Algorithm-aided Information Design: Hybrid Design approach on the edge of Associative Methodologies in AEC

Evgenii Ermolenko

BIM A+

UMinho | 2020

Universidade do Minho
Escola de Engenharia

Evgenii Ermolenko

Algorithm-aided Information Design: Hybrid Design approach on the edge of Associative Methodologies in AEC

BIM A+ European Master in
Building Information Modelling

Co-funded by the
Erasmus+ Programme
of the European Union

September 2020

Universidade do Minho
Escola de Engenharia

Evgenii Ermolenko

Algorithm-aided Information Design: Hybrid Design approach on the edge of Associative Methodologies in AEC

BIM A+ European Master in Building Information Modelling

Master Dissertation
European Master in Building Information Modelling

Work conducted under supervision of:
Bruno Figueiredo
Rui Dias

Co-funded by the Erasmus+ Programme of the European Union

September, 2020

ACKNOWLEDGEMENTS

First and foremost, I must acknowledge with gratitude the contribution of the thesis supervisor Bruno Figueiredo who guided the study process with a constant research review and scientific sources support. His patience, encouragement and insightful editorial direction in difficult circumstances helped me to achieve the final results.

I would like to thank the partner company NOZ Arquitectura in the person of Mr. Rui Dias for their support in the proposition and supplying of case studies for the dissertation.

Besides, the significant influence was gained from other researchers such as Mark Burry, Branko Kolarevic, and Harri Humppi. Their scientific researches especially contributed to the theoretical provision basis of the present study, which endeavours to extend further boundaries of their consolidated efforts to investigation in the field of Digital Design. Likewise, I gratefully acknowledge all the professionals pointed out in the reference list.

In addition, I woud like to thank LLC “ProruBIM”, the company based in Moscow, providing professional BIM services, for their generous support by supplying the DynoStudio, the tool for elaboration of UI for Dynamo scripts developed during the case studies.

Finally, I would like to express gratitude to the consortium of the BIM A+ Erasmus Mundus Master Program for their technical assistance and cooperation while writing the present thesis.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

[illegible]

RESUMO

As últimas três décadas trouxeram um progresso colossal às metodologias de projeto, numa evolução em direção a uma fusão entre os mundos digital e físico que se expande pelo poder da computação e redes digitais. Neste histórico período, surgiram duas gerações de metodologias e ferramentas no contexto do CAD: sistemas de geração Aditivos e sistemas de geração Associativos. Atualmente, projetistas de todo o mundo dedicam-se a novas formas de exploração do projeto. Neste processo emergiram duas metodologias a partir da abordagem de Projeto Associativo – Object-Oriented Design (OOD) e Algorithm-Aided Design (AAD).

O objetivo principal deste trabalho é investigar, examinar e explorar os limites de OOD e AAD para a num novo paradigma de projeto, onde as vantagens de ambos os métodos se combinam numa nova metodologia generativa que no presente estudo se design de Algorithm-aided Information Design (AID).

A metodologia deste estudo estrutura-se em dois momentos. No primeiro momento, metodologias CAD existentes são investigadas e a estrutura conceptual é extraída com base na análise do estado da arte, em seguida, os dados analisados são sintetizados na proposta do tema central do trabalho. No segundo momento, ferramentas e processos de trabalho são elaborados e examinados na prática para confirmar a proposta do tema.

Em conformidade, o conteúdo da investigação é composto por duas partes teóricas e práticas. Na primeira parte teórica, é constituído por uma revisão da literatura e hipóteses para especular sobre a metodologia AID, suas ferramentas, possíveis vantagens e desvantagens. Em seguida, pelo desenvolvimento de casos de estudo realizados de acordo com estágios sequenciais de projeto digital à luz da implementação prática da metodologia AID.

Os casos de estudo abordam diferentes aspetos de projeto, tal como a geração de modelos e documentação, automatização de processos de desenho, interoperabilidade, controlo da fabricação, análise e otimização do desempenho do projeto.

Finalmente, desenvolve-se um conjunto de projetos que implementam diversos aspetos da metodologia AID. Após esta parte prática, a investigação retorna à teoria onde as informações analíticas são reunidas com base na revisão da literatura, quadro conceptual e relatórios da prática experimental. Associando e analisando todas as fontes investigadas, o estudo apresenta proposta de melhorias da estrutura do AID e suas conclusões. Em resumo, o estudo sintetiza a metodologia AID como parte de um processo de Desenho Híbrido, possibilitando o uso criativo de ferramentas e a elaboração de sistemas de projeto Ágil integrando metodologias aditivas e associativas de Projeto Digital.Em geral, o estudo é baseado no método Ágil e em métodos de desenvolvimento cíclico misto entre prática e teoria para alcançar uma visão ampla do tema.

Palavras chave: DAA (Desenho Assistido por Algoritmos), Desenho Associativo, DIA (Desenho e Informação Assistidos por Algoritmos), DOO (Desenho Orientado a Objetos), MIC (Modelo de Informação para a Construção)

ABSTRACT

Last three decades have brought colossal progress to design methodologies within the common pursuit toward a seamless fusion between digital and physical worlds and augmenting it with the of computation power and network coverage. For this historically short period, two generations of methodologies and tools have emerged: Additive generation and parametric Associative generation of CAD. Currently, designers worldwide engaged in new forms of design exploration. From this race, two prominent methodologies have developed from Associative Design approach – Object-Oriented Design (OOD) and Algorithm-Aided Design (AAD).

The primary research objective is to investigate, examine, and push boundaries between OOD and AAD for new design space determination, where advantages of both design methods are fused to produce a new generation methodology which is called in the present study AID (Algorithm-aided Information Design).

The study methodology is structured into two flows. In the first flow, existing CAD methodologies are investigated, and the conceptual framework is extracted based on the state of art analysis, then analysed data is synthesized into the subject proposal. In the second flow, tools and workflows are elaborated and examined on practice to confirm the subject proposal.

In compliance, the content of the research consists of two theoretical and practical parts. In the first theoretical part, a literature review is conducted, and assumptions are made to speculate about AID methodology, its tools, possible advantages and drawbacks. Next, case studies are performed according to sequential stages of digital design through the lens of practical AID methodology implementation.

Case studies are covering such design aspects as model & documentation generation, design automation, interoperability, manufacturing control, performance analysis and optimization.

Ultimately, a set of test projects is developed with the AID methodology applied. After the practical part, research returns to the theory where analytical information is gathered based on the literature review, conceptual framework, and experimental practice reports. In summary, the study synthesizes AID methodology as part of Hybrid Design, which enables creative use of tools and elaborating of agile design systems integrating additive and associative methodologies of Digital Design.

In general, the study is based on agile methods and cyclic research development mixed between practice and theory to achieve a comprehensive vision of the subject.

Keywords: AAD (Algorithm-Aided Design), AID (Algorithm-aided Information Design), Associative Design, BIM (Building Information Modelling), OOD (Object-Oriented Design).

TABLE OF CONTENTS

This page is intentionally left blank

1. INTRODUCTION: THEORY BASICS

1.1. SUBJECT OF RESEARCH

In this dissertation, a sphere of research lies between the Algorithm-Aided Design (AAD) and Object-Oriented Design (OOD). AAD and OOD are currently the most prominent design approaches in practical AEC industry as well as theoretical. Consequently, a quite extensive investigation has been dedicated to both of them, and much scientific literature and applied primers have been written. From these investigations, two modelling methodologies have arisen: Algorithm-Aided Modelling (AAM) and Building Information Modelling (BIM) within AAD and OOD concepts, respectively.

The next development stage is seen in its closer integration loop. Evidentially a growing number of enthusiasts in the academy and industry are elaborating and implementing new workflows utilizing both design methodologies. Nonetheless, they are not systematic and mostly project-specific solutions yet. The benefits of possible joint design workflows between BIM and AAM remained depreciated worldwide until recent years, and it still requires extensive research for more comprehensive active implementation. Among potential advantages of such combined workflows, could be enhanced control and accuracy of geometry and datasets, computational aid integration, direct design-to-fabrication processes, automation, and agile design techniques.

1.2. OBJECTIVES

The dissertation focus is on a critical investigation and examination of boundaries of dominant design methodologies, BIM and AAM, in pursuit to determine the space of possible interactions within shared workflows and to define a holistic AIM methodology on the edge of both of them. In other words, the main goal of the study is an exploration of the applied techniques combining AAD ann OOD tools within the design workflows and development of a methodology to reveal new potential advantages of symbiotic processes in digital design. The developed methodology should be applied in practice part with the participation of the partner company. Along with the intent of getting to grips of the main subject, there is aim to increase implicit knowledge about cutting-edge digital design methods, tools, and theories that stand behind them. In this respect, the dissertation is perceived as a means to enhance professional adequacy. The result of the research is presented in the Final Conceptual Framework in Summary Chapter. Though it is hard to embrace all aspects within the reserved format. Therefore, the presented framework is generalized for clarified delivery purpose. Ultimately, the dissertation contributes to a leaner understanding and implementation of discovered methods within the AEC industry. Following aforesaid, next objectives are pointed out from general to more specific:

- Contribute to a systematic approach and leaner understanding of Algorithm-aided Information Modelling and Design
- Increase personal knowledge about cutting-edge digital design methods, tools, and theories stand behind them
- Enhance professional adequacy with theoretical, modelling, scripting, manufacturing skills and developing a critical vision on their application
- Develop fruitful collaboration with the architectural design company and integrate AID methodology with developed tools
- Find particular solutions for design problems and apply them
- Solve ubiquitous and tedious tasks with AIM methodology specific for site data acquisition, performance analysis and simulation, documentation authoring, manufacturing, and form-finding. Besides, the test of verification and communication tasks during design development

1.3. INTRODUCTION TO THEME

1.3.1. A BRIEF HISTORY OF CAD.

AEC Industry has drastically evolved over the last decades of the digital age. Architectural design methodology changed from the digitalizing drawing board to generation of information systems. Ultimately, a new approach of modelling associative information systems with controlled algorithmic logic has been established.

However, this race began long earlier than PC emerged. Before digitalization, the dominating tool for idea representation was paper drawing. Despite the limitations, drawings have been the stable medium of architecture over the centuries, and this was possible as architects have mostly relied on typology, i.e. the use of well-proven, preconceived solutions, and tectonic systems. (Tedeschi, 2014) Until the end of the 19th century, a comprehensive typologic and stylistic database was cumulated. However, in the late 19th century with the industrial revolution and intensive urbanization, new challenges and technics raised. Functionalism and optimization became priorities. Consequently, the conventional drawing was first attacked by a new approach, the form-finding – which aimed to investigate novel and optimized structures found through complex and associative relations between materials, shape, and structures. (Tedeschi, 2014). Pioneers like Gaudi (1852-1926), Frei Otto (1925-), and Musmeci (1926-1981) have rejected existing typology and looked to self-formation processes in nature as a way to organize buildings. They used physical models and computational methods to demonstrate how dynamic forces could mould self-optimized architectural forms. Structural optimization through physical modelling was mostly mono-parametric (gravity-based). It marked a trajectory towards multi-parametric form-finding and responsive architectural model as an associative information system which aims to interact with heterogeneous data: geometry, dynamic forces, environment, material, economic, social data. Over the last century, the increasing complexity of buildings has made form-finding an essential strategy in determining the shape and organizing architectural spaces.

Luigi Moretti, the Italian architect, invented the definition for "Parametric Architecture" in 1939. He said: "The parameters and their interrelationships become the code of the new architectural language, the "structure" in the original sense of the word. The setting of parameters and their relationship must be supported by techniques and tools offered by the most current sciences, in particular by logics, mathematics, and computers. Computers give the possibility to express parameters and their relations through a set of (self-correcting) routines". (F.Bucci and M. Mulazzani, 2006).

Later, Douglas C. Englebart, in his paper "Augmenting Human Intellect: A Conceptual Framework" in 1962, asserted the vision for the future architect, suggested object-based design, parametric manipulation, and architectural model as information system maintained with a relational database.

1.3.2. FROM ADDITIVE TO ASSOCIATIVE DESIGN

As seen above, the concepts well-known nowadays such as CAD (Computer-Aided Design), BIM (Building Information Modeling), Computational, and Parametric Design had been developing for roughly a century since times before computerization. So, when the first design application, SketchPad, utilizing the computer with a visual interface occurred in 1963, it already used raw parametric constraints and associative logic, the so-called atomic constraint. Together, the introduction of the computer and parametric architecture concept by Moretti, and the graphical interface of SketchPad by Ivan Sutherland marked a revolution in architectural design methodology. It upgraded design tools as well. However, the innovations brought by early CAD programs were not immediately embraced by commercial software for almost three decades. (Tedeschi, 2014) Commercially successful software, such as AutoCad (1982), met the architects need to speed up repetitive tasks and manage multiple drawing layers, by in effect, digitalizing the drawing board.

In essence, such software utilizes the "first-order" or Additive digital technique including direct CAD commands for drawing polylines or splines using mouse-clicks, and usually referred to as "hand-drawn". "Second-order" or Associative techniques assert modeller instead to operate on an instruction set – algorithm or parametric system – that in turn produces design geometries and related metadata. The designer develops an associative set of rules and processes - or "procedural model" – and then feed the system with input data. Then, input data is translated and transformed through procedural environment controlled by algorithms defined by the designer or vendor, and the system produces geometrical and meta-data output. (Stasiuk, 2018)

Only in the late 1980s, two prominent concepts branched out further from the same origin of Associative Design – OOD (Object-Oriented Design) and AAD (Algorithm-Aided Design).

OBJECT-ORIENTED DESIGN: BUILDING INFORMATION MODELLING

The first one has developed into Building Information Modelling. This concept intended to create building digital twin consisted of parametric components supported by the integrated relational database and associative logic in modelling software. Perception of the building as a database contributed to the breakdown of architecture into its constituent components, necessitating a literal taxonomy of building subelements. One of the first projects successfully implemented building database was the Building Description System (BDS) created in 1975 by Charles Eastman. BDS was the first software to describe individual library elements that can be retrieved and added to a model. This program already had a Graphical User Interface (GUI), orthographic and perspective views, and a sortable database that allowed the user to retrieve information categorically by attributes including material type and supplier. (Quirk, 2013) Then, in 1987, ArchiCAD was launched, as the first BIM software available on a PC. (Bergin, 2011) Nowadays, Building Information Modelling, as part of Object-Oriented Design methodology, has become an essential tool in the development of the contemporary AEC industry.

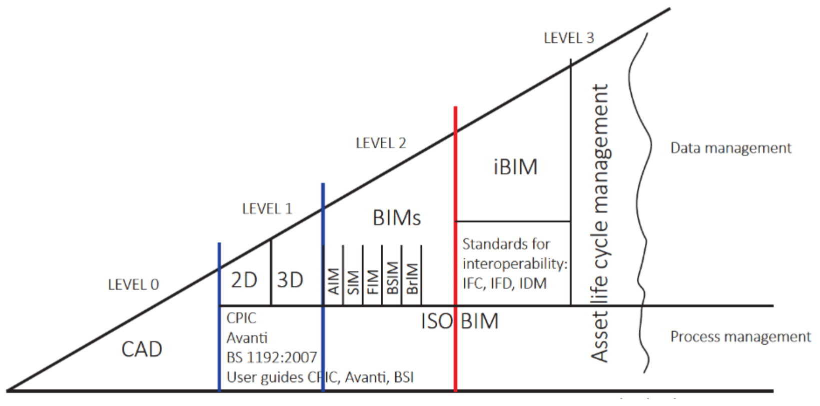


Figure 1.1 - Maturity Diagram of BIM

The Maturity Diagram of BIM. Level 0 - 2D CAD. The diagram shows that BIM can be divided into Data and Process managements which contain common BIM standards and processes. Level 1 utilizes both 2D and 3D information in projects. On Level 2 all disciplines use their own 3D models. The collaboration is based on information exchange between disciplines by using common file formats such as IFC. On Level 3 all disciplines work with the same shared project model. The risk for conflicting information disappears. (RIBA, 2012, p. 3.) Modified from (RIBA, 2012, p. 3).

(Architecture, Engineering & Construction). BIM tools and methodology correspond to media age challenge overall.

DRAWBACKS OF BIM

Nevertheless, regardless of all endeavours with rapidly developing and enhancing quality and interoperability of BIM, AEC is still a less digitalized industry, among others. Currently, the BIM experiences the Second Level of maturity throughout its development, according to the RIBA statement, 2012. (Figure 1.1) This stage is distinguished by collaborative work and requires "an information exchange process which is specific to the project and coordinated between various systems and project participants" (Scottish Futures Trust, as cited in Richard McPartland, 2014). BIM methodology has been successfully applied in many companies worldwide and managed collaboratively through Common Data Environments (CDE). However, it is a verified information bridge gap mainly due to proprietary software solutions, current limitations of common IFC format, and lack of internationally accepted standard support, despite ongoing development efforts. As a result, relevant information loses its value throughout the project lifecycle. Therefore, precise workflow coordination and customizing tools are vitally important to the current maturity level of BIM.

Besides information exchange issues, other technical problems are interrupting the design workflow as well. Meanwhile, the complexity and functionality of BIM grow, the technical part of design within disciplines also becomes more complex and requires even more time than design itself. (Dias, 2020) Technical part includes but is not exclusive to data acquisition/capturing, performance analysis & simulation, project documentation, verification, data management, and fabrication. Most of these tasks solutions require narrow technical knowledge and pre-defined workflows, whilst the design part still much more human dependent and stipulates time-consuming and brain-storm analysis. For the more effective design process, technical routine shall be identified and automated to liberate resources for creative tasks.

ALGORITHM-AIDED DESIGN: ALGORITHM-AIDED MODELLING

The second concept has been focused on Algorithm-Aided Design (AAD). The term defines the use of specific algorithms-editors to assist in the creation, modification, analysis, or optimization of a design. Arturo Tedeschi introduced the definition in 2014 in his book "Algorithms-Aided Design, Parametric Strategies using Grasshopper". Employing Object-Oriented Programming, the designer creates algorithms which set associative rules controlling model and data flows. AAD implies the operating on parametric system interactively controlling model as system output. Such output is defined in the study as System Output State (SOS) because the model reflects the state of the system at certain input conditions. Even though Object-Oriented Programming was a part of BIM since the early years, there was a limited library of elements and methods for use, prescripted by vendors. However, the original concept of AAD implies associative logic controlled by end-user. The most profound progress has happened from the late 1980s to the present day. Academic research and Avangard practices – trying to escape simple editing limitations of software – explored new ways to manipulate models aiming to find unexplored solutions and forms through programming. Many designers soon realized that more sophisticated programs could manage complexity beyond human capabilities by structuring routines into procedures. This type of modelling relies on programming languages that express instructions in a form that can be executed by the computer through a step-by-step procedure – the algorithm. (Tedeschi, 2014) Algorithm programming is performed in a specific Graphical Algorithm Editor (GAE) which can be a standalone application or embedded in a software application. For instance, standalone editors include Python, C#, C++, etc. Embedded editors are provided by programs such as Rhinoceros, Maya, Revit, Autocad. These editors allow the use of Application Programming Interface (API) to manipulate model, create datasets and computational geometry as well as script encode relationships over them. In other words, objects are no longer controlled with a mouse. Instead, they are defined by procedures expressed in a specific program language. There are two types of program languages: Visual Program Language (VPL) and Textual Program Language (TPL). Both types can build procedural algorithms in GAE either separately or mixing within a single script. Though programming push designer capabilities up and allows to manage complex information systems followed with the model, AAD has been developing separately from BIM for over a decade. It has been more present in the form-finding and conceptual design stage within 3d modelling software such as Rhinoceros.

Meanwhile, AEC specialists have been struggled to manage the growing complexity of building information systems during the authoring design stage. The potential of Algorithm-Aided Design was hidden for a while. In 2000s AAD tools seamlessly began interring the AEC industry by introducing new object-oriented languages such as GDL and RhinoScript. Then appropriate GAEs (Graphical Algorithm Editors/ Visual Programming Interfaces) such as GenerativeComponents for ArchiCAD and Grasshopper for Rhinoceros emerged almost at the same time in 2007. Ultimately, it was a colossal leapfrog encouraging designers worldwide to explore and speculate design by digital means in a different way. However, early AAD tools still grew in parallel with BIM tools. A large amount of literature of 2000s dedicated to both concepts existing separately without mentioning their possible integration (with few exclusions) is evidence of the former state. In my opinion, the game changed only in 2012 when DesignScript was integrated as an associative language providing support for BIM authoring tools, their classes and methods. DesignScript combines Visual Programming and Textual Programming (see Glossary) within joint Integrated Development Environment (IDE) including GAE. Currently, it runs as the computational engine within Dynamo Studio and integrated with Revit, the BIM authoring software.

DesignScript is intended to encourage the integration of different design disciplines and to support projects of different size and complexity by introducing Scalability between different levels of scripting. (Aish, 2013) At one level DesignScript is an intuitive application which can be used by designers as novice programmers with the minimum programming prerequisites. On another level, behind this intuitive user interface, it is a highly innovative programming language introducing a number of domain-specific programming ideas including associativity, replication and modifiers. These innovative ideas are combined with well-established conventions drawn from imperative, functional and object-oriented programming languages and unified into a single scalable computational design application. A special characteristic of the user interface is that it supports the progression from novice user to more accomplished programmer by progressively revealing its capabilities. (Aish, 2013)

Consequently, the use of AAD within a BIM environment increased among architects and engineers, affecting the several stages of the design and construction stages. (Silver, 2006; Burry, 2011) The design brief for a building project is commonly described as a series of constraints, parameters, databases attached, and interacted with the 3D model. While the translation of this data into a design occurs manually across plenty of software with information loss, there are benefits to transfer this process more explicitly and systematically, through the application of scripts and parametric systems, directly throughout the whole life cycle of the project. Since primary AAD tools were elaborated in the late 2000s, extensive development of assessory addons and plugins have been activated. Such a mainstream was caused particularly by the open-source basis of AAD platforms, internet expansion, and a large, engaged community of developers and designers. These tools integrate with the CAD and BIM software through Application Programming Interface (API) embedded within the authoring environment that ease the implementation of scripting processes by AEC professionals. Rapid dissemination of these processes should result in a combined approach that utilizes both BIM and AAD processes from generation and assessment of conceptual design solutions, through performance simulations, managing information flows, and fabrication.

DRAWBACKS OF AAM

Main drawbacks of AAM comparing to BIM are around the lack of element classification and semantic data. AAM systems consist only of geometry entities such as points, curves, surfaces, and solids. Thus, systems don’t contain complex metadata attached to parametric elements, unlike objects in object-oriented design software. In other words, AAM system does not include database dimension and material-based knowledge associated with digital models (Zarzucki, 2012, p. 1). As a consequence, AAM systems seem inadaptive to BIM environment as they cannot carry out and translate relational data within a single workflow except geometry itself. Necessary data exchange process from AAM to BIM software breaks the link between the model and its generating script. (Boeykens, 2012) Consequently, the geometric System Output State becomes static and non-parametric after export to external tools. To update its version, it is required to export model again and substitute instead of the former one.

1.4. CONCEPTUAL MODEL

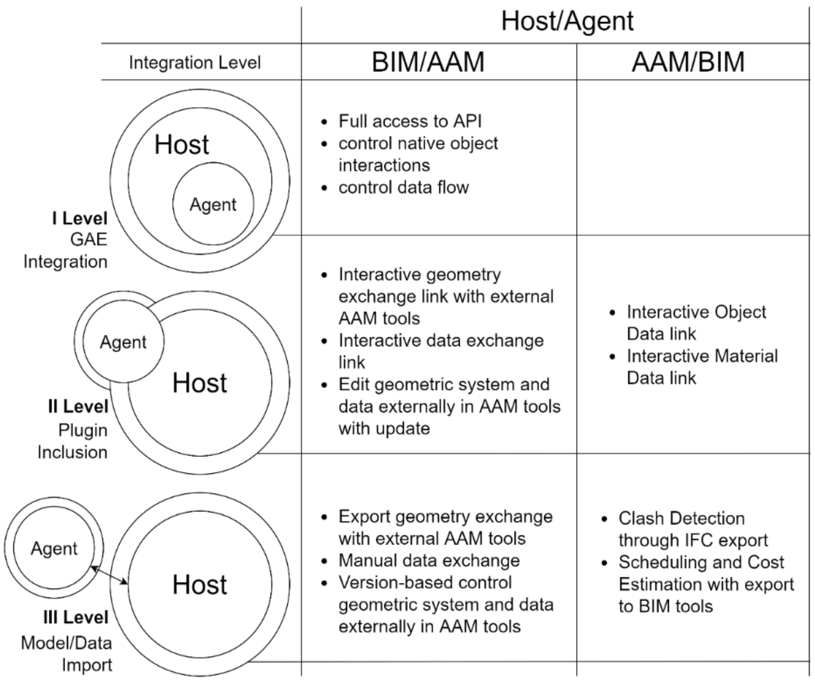
As aforementioned, the subject of research is focused on space between AAD and OOD approaches and on the combination of their methodologies. In order to nourish this space with new techniques or an entirely new methodology, research requires relationships type determination first of all. The study considers such relationships either as assisting or symbiotic. Assisting relationships could be segregated into three subtypes: complete Graphical Algorithm Editor (GAE) Integration, Plugin Inclusion, and Model/Data Import. Each subtype corresponds to Integration Level from one to three with integration downgrade. Integration Levels present characteristics of assisting Host/Agent relationships in Figure 1.2. Another type of Peer-to-peer Adoption is presented in detail in Figure 1.2.

First Integration Level is peculiar to BIM hosting systems. It implies Graphic Algorithm Editor to be installed into BIM software. In this Level, single-source BIM model is controlled both by the procedural system defined in GAE and through Graphic User Interface (GUI) built-in with BIM platform. Distinctions of such type are the single model, two interfaces with access to the same native objects through Application Programming Interface (API) and GUI, minimum information transfer losses and controlling data flow directly in BIM software. Dynamo shows the first significant steps that are taken in this direction. However, development is still undergoing to enhance the integrity of the design process. So Additive and Associative tools can be used simultaneously without the need to translate objects between editor environments to switch the tools. Probably this unification can eventually lead us to a new approach to digital design.

Second Integration Level can be applied both to BIM and AAM hosting systems. In this Level, Host system exchanges data with external Agent tools interactively through the plugin that links systems. In the case of BIM hosting system, the interactive data flow is controlled by the Agent procedural system. Special code block in the procedural system translates data to object-oriented modelling environment with the semi-manual designation of object classification information and metadata attached. Ubiquitous plugins for interactive linking AAM and BIM environments on the time of study conducting are Rhino.Inside, Speckle, and GRevit. (refer to Glossary, Tools) Rhino.Inside allows direct access to Grasshopper GAE and procedural system definition as well as to Rhino GUI and geometric model right from Revit and translates this data interactively running the procedural system through Grasshopper GAE and Rhino GUI. Meanwhile, Speckle streams the procedural System Output States (SOS) online through the cloud and allows to convert them to native objects in Revit automatically. Grevit sends the information in the same way as Speckle does but only in one direction from Rhino to Revit. In opposite case scenario with AAM hosting system, it interprets the object information for representation, analysis, or simulation directly from BIM model. Such information can be object classification, manufacturing, geometric, or material data. It notable though that only geometric and material data can be translated into AAM model of SOS. The same plugins are used for linking BIM environment into AAM.

Finally, the third Integration Level is also representative of BIM as well as AAM hosting systems. This Level states for export/import data through external formats such as SAT, FBX, DWG, CSV, and GEO. The export can be manual or automated through procedural system definition. Add-ons are used for automating export of data. Some of the most well-known of them are Rhynamo, Mantishrimp, Hummingbird, and BIM GeomGym IFC. (refer to Glossary, Tools) Rynamo reads geometry from Rhino file and converts to Revit Families semi-manually in Dynamo. Mantishrimp exports geometry, data to .geo file from Grasshopper, reads it and converts to Revit Families manually in Dynamo. Hummingbird exports points, curves, or parameters to CSV file from Grasshopper and running this file from Revit to construct families automatically. BIM GeomGym IFC enables IFC (Industry Foundation Class) model to be generated and exchanged from AAM model of SOS to ArchiCAD, Revit, Bently, Tekla and any other BIM software with IFC capability.

In contrast to Assisting relationships typology, Symbiotic or Peer-to-peer type implies a group of independent hosting systems interactively exchanging data. Every system has its own specific utilization. BIM system can be used for 3D coordination & validation, asset management, collaboration, phase planning, scheduling, and cost estimation. Meanwhile, AAM System can be applied for form-finding, generation & management of complex geometry, simulations, analysis, optimization, and manufacturing. In another way, systems can be deployed according to the design part. For instance, shell structure can be hosted by AAM system. Meanwhile, interiors and room entities are controlled by BIM system. The important note is that such relationship necessitates Geometric and Information Source hostage shall be strictly defined to avoid alterations version conflicts. Coordination System (CS) is applied to manage such relationships. It can be a system of any type created by any method. Besides geometry coordination, CS enables management of Design and Solution Spaces with sufficient constraints which can be dynamic input parameters or computed parameters. Valid source system deployment contributes to the accurate and effective design process. Ultimately, Symbiotic relationships type allows interactive, reliable, and optimized workflow with constant data exchange and rational deployment of information between systems. (Figure 1.2)



	Host/Agent	
Integration Level	BIM/AAM	AAM/BIM
I Level GAE Integration	• Full access to API • control native object interactions • control data flow	
II Level Plugin Inclusion	• Interactive geometry exchange link with external AAM tools • Interactive data exchange link • Edit geometric system and data externally in AAM tools with update	• Interactive Object Data link • Interactive Material Data link
III Level Model/Data Import	• Export geometry exchange with external AAM tools • Manual data exchange • Version-based control geometric system and data externally in AAM tools	• Clash Detection through IFC export • Scheduling and Cost Estimation with export to BIM tools

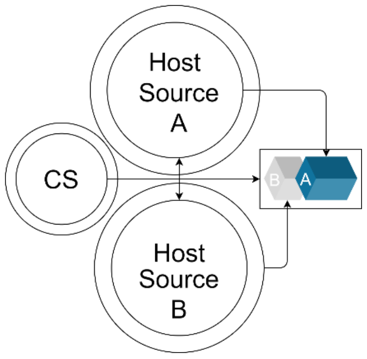


Figure 1.2–Assisting BIM/AAM Relationships Typology Table (left); Symbiotic Relationship scheme (right)

1.4.1. AIM Conceptual Model

Regardless of relationships type, becoming deeper interactions between AAM and BIM environments result in a new methodology. As expected, it fuses both methods into shared workflows. However, the study proposes this methodology as next-generation, holistic and inherent from AAM and BIM. The holistic approach means that methodology is not just a summary of former methods but rather deduces new characteristics to the system. Such methodology compensates drawbacks of its predecessors and allows playfulness with a wide variety of possible workflows within the aforementioned relationships. Among unique aspects of this methodology should be a seamless data flow, free computational form-finding and form-making, direct design-to-fabrication workflow, control over proprietary interfaces with the development of custom tools, automation of routines. The research suggests the definition of this methodology as Algorithm-aided Information Modelling (AIM). The design utilizing this methodology is Algorithm-aided Information Design (AID), respectively. The conceptual model of AIM is presented in Figure 1.3.

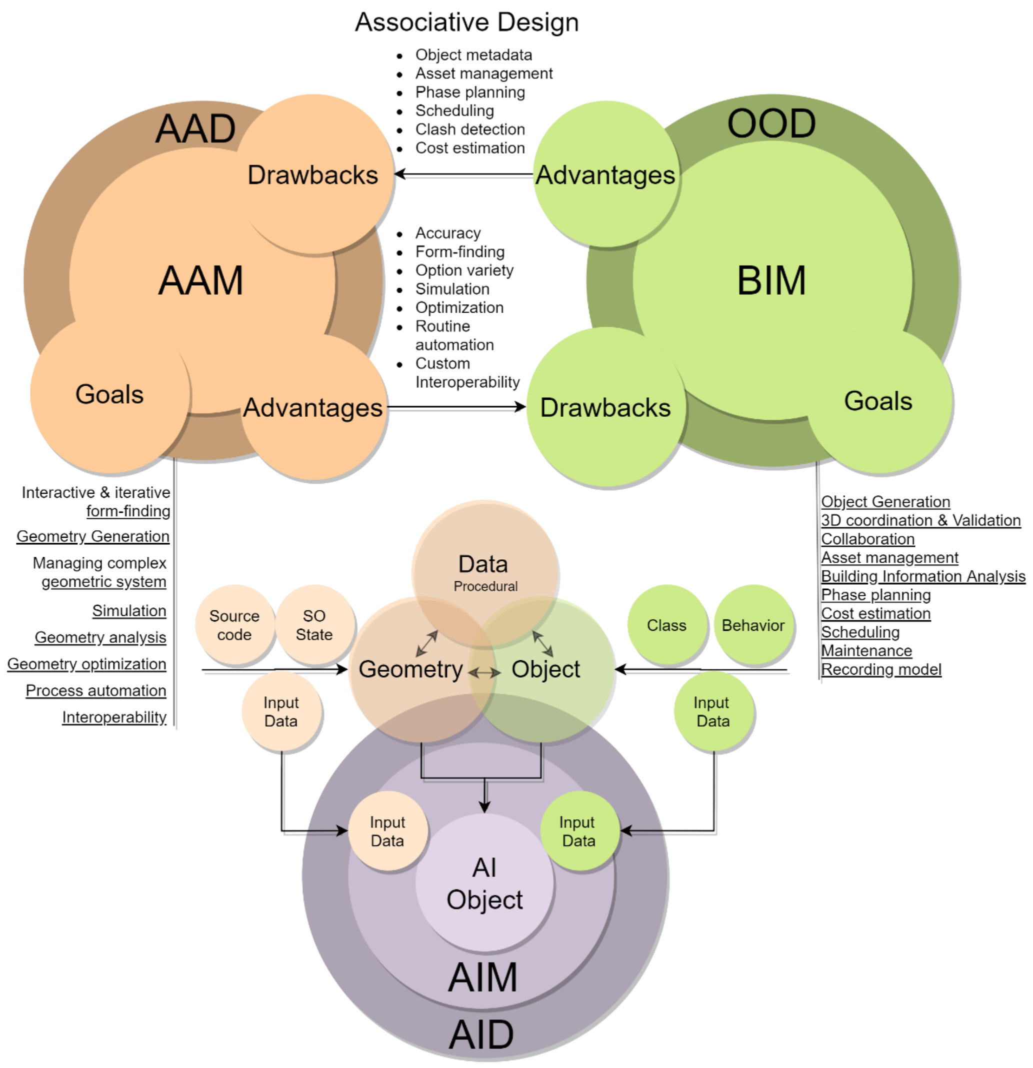


Figure 1.3–Algorithm-Aided Information Modelling Conceptual Model

1.4.2. CONCEPTUAL FRAMEWORK

Since OOD and AAD emerged and began to influence one another, plenty of accessary concepts and techniques have been deployed. Consequently, the degree of "relaxation" in the use of interdependent terminology enhanced enormously for the last two decades. Hence, the study necessitates building a framework from the scope of definitions related to digital design for being coherent and for comprehensive image of subject. (Figure 1.4)

First, two types of digital design are distinguished due to their nature – Additive and Associative. As mentioned above in section 1.3.2, Additive design is regarded as "first-order" design which means that model is manipulated directly by mouse-clicking. "First-order" techniques are used in polygonal and NURBS-modelling software such as Rhinoceros and SketchUp. In contrast, Associative design is considered as "second-order" design which assumes that the model is managed by algorithms. In turn, algorithms are controlled by the designer. In other words, while the additive model is a product of the "first-order" digital techniques and produced semi-manually, the associative model is based on translational and transformative interactions with the designer through a relational algorithmic system. (Reilly, 2018) Meanwhile, algorithms can be encoded into predefined methods written in software libraries which in turn can be accessed and used in custom algorithms through Application Programming Interface (API) or can be written directly by the designer. In this fashion, parametric methods also refer to Associative Design as any parametric model is algorithmically defined and controlled through manipulating parameters. Subsequently, except of Algorithm-Aided Design (AAD) itself, Object-Oriented Design (OOD), and Geometry-based Parametric Design (GPD) are also covered by Associative Design definition.

OOD is based on object classes with predefined methods and properties which prescribe the behaviour of objects. Moreover, API in BIM software such as Revit enables access to classes and their methods. So they can be used in custom algorithms as well. GPD is also referred to Associative Design because it uses predefined modifiers for model manipulation (e.g. Modifier List in 3DsMax). Altering input parameters, we control modifiers which effects on model output. Though, both BIM and GPD software leave the ability for the direct mouse-clicking manipulation as well.

Associative Design methodologies allow creating interactive parametric systems. Thus, these methodologies can be used as means of Procedural Modelling. "A Procedural Modelling process is one that uses an explicit instruction set to produce a model outcome." – as defined by David Stasiuk, who developed a graded framework for thinking about Procedural Modelling systems. He aimed to segregate procedural system types, identify the difference and facilitate a categorical understanding of these systems and their diverse functional capacities. In the context of Procedural Modelling, all the models are indeed the output states of algorithmic systems. Therefore, from now on, the definition "system" will be used for referencing to associative and procedural design models. Meanwhile, System Output State (SOS) is used instead of Model as it better reflects algorithmic system dynamic nature. From Procedural Modelling, Stasiuk distinguished three consequential system types: parametric, computational, and generative. Each type covers definitions from its predecessors in sequence hierarchically. Further description is synthesized based on different sources with the main accent to Stasiuk' framework with assumptions from my personal experience.

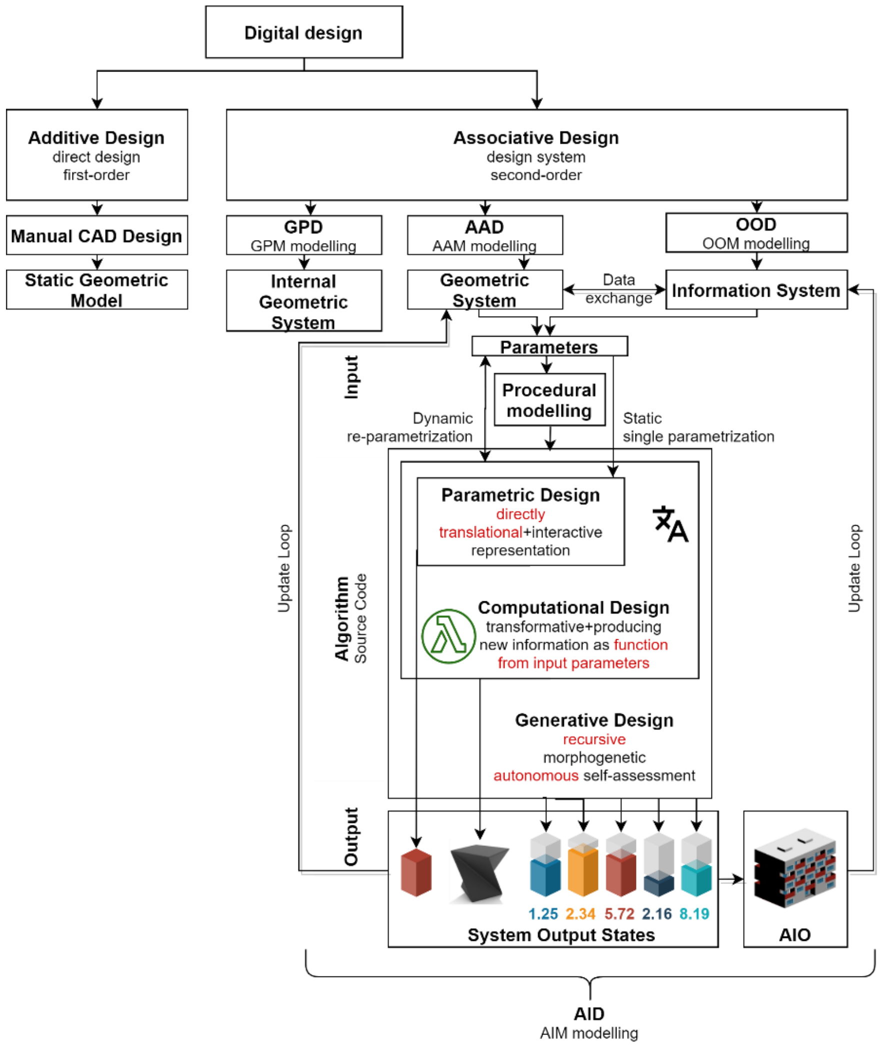


Figure 1.4 – Digital design **Framework**

The Parametric System is defined as "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) A parametric system is composed of parameters, translational functions, and their expressed information, including geometrical and metadata. In other words, these three primary parts of the system are input parameters, algorithms and System Output States. The parametric system is functioning as a predetermined conversion of input parameters to System Output State in only translational and in no transformative way, for procedural representation purposes. Meanwhile, no new information is generated. In its most basic type, a parametric model operates as a one-way system, with parameters feeding an algorithm that directly enacts outcomes through explicit transformation. Once the designer has determined this explicit setup, his role shifts from design to system operation. Schumacher said: "While the attributes of the graphic/digital primitives […] are fully determined and fixed at any time, within the parametric diagram they remain variable. This variability might be constrained within a defined range on the basis of associative functions that imbue the diagrammatic process with an in-built intelligence". (P.Schumacher, 2010)

As Robert Aish claimed in 2011, parametric systems are deployed to significant effect through the representational capacities specific to BIM systems. In this capacity less as form generators or response systems for adaptation, parametric systems demonstrate excellent capability for information collection and collation, and as power drawing instruments.

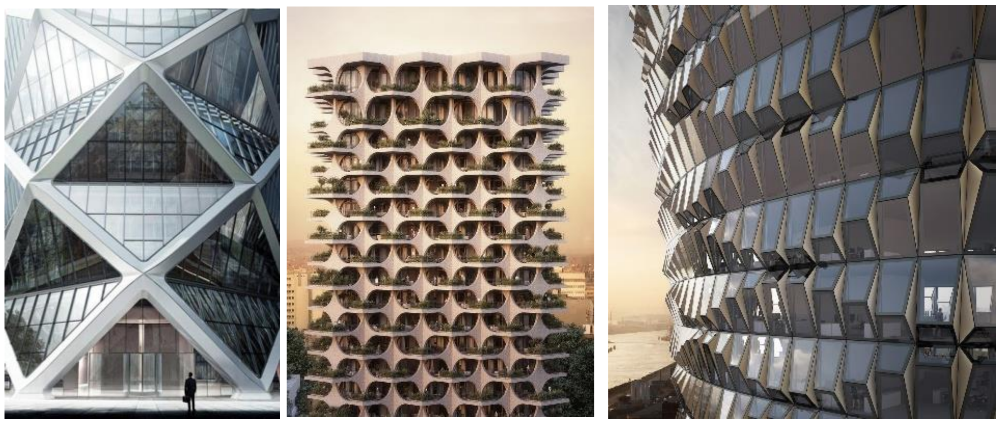


Figure 1.5 – Architectural Projects with applied Parametric Design, from left to right: Poly International Plaza by SOM, Beijing; Residential tower by Precht Architects, Tel Aviv; façade parametric system by Schuco

Computational Systems are operated according to "a set of procedures consisting of a finite number of rules, which define a succession of operations for the solution of a given problem." (Ahlquist, 2011) The essential distinction is that a Computational System is able of transforming input data into new data by functional criteria. In other words, it deduces the System Output States based on input parameters and algorithmic procedures utilizing functional dependencies, unlike the Parametric System, where input parameters are directly translated into the System Output State. Another distinction is that a Computational System has "the capacity for parameters to be actualized as dynamic and responsive entities during system execution, not only informing the algorithm but in turn being recalibrated themselves" – as was stated by David Stasiuk. In contrast, a Parametric System exploits parameters as single static entities during execution.

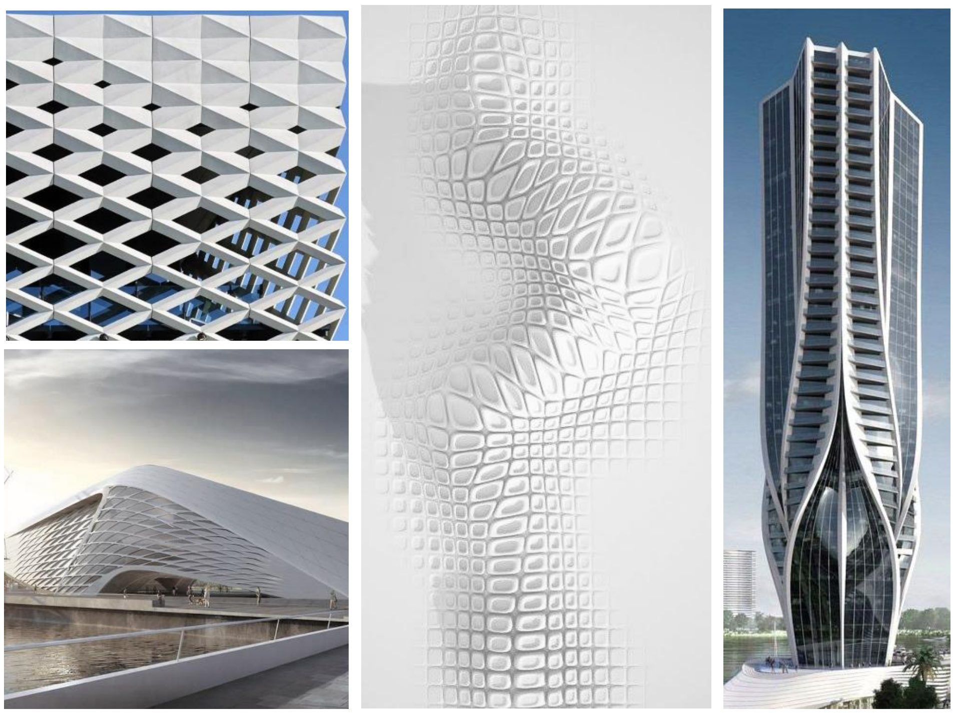
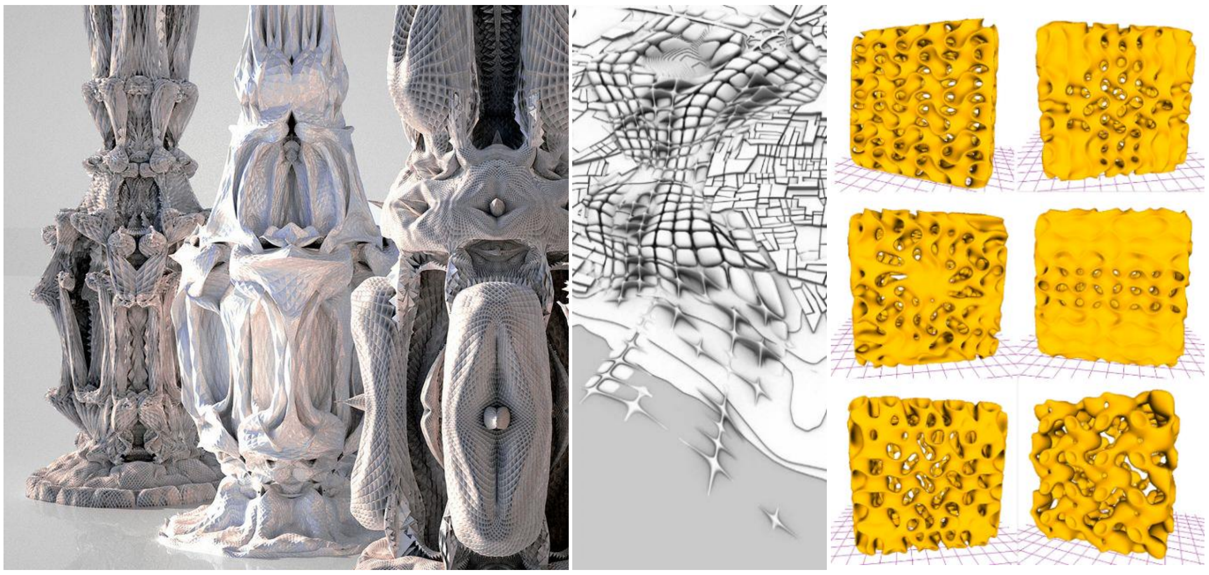


Figure 1.6 – Architectural Projects with applied Computational Design, from left to right: shopping mall façade by ACME, London; adaptive fabric, author is anknown; residential tower by ZHA, Brisbane

Eventually, Generative System is defined as a procedural system with certain algorithmic autonomy which allows deploying algorithmic transformations and "re-use embedded sub-systems" for specifically incremental morphogenesis. (G.S. Hornby, 2001). In compliance, the Generative process is considered as "an active space of progressive formation and mutation" in the digital design process. (Attar, et al 2009) In other words, Generative Design is seen as an iterative process that allows generating autonomously multiple design options within Solution Space, and, using advanced algorithms, to determine the best solution. The designer role in this workflow is to set design goals, define constraints, write a recursive algorithm which will be used for transformations iteratively generating multiple design options, and finally to evaluate outputs and select the best design option. (Figure 1.7, Generative Design workflow, bottom graph) Thus, the control of design in Generative System has shifted from the design solution itself to design process. Constraints of the process control the generation of design solutions. (Sevaldson, 2005, p. 16.) Consequently, the computer is seen as a tool that produces alternative design solutions that can be evaluated by the designer.

All in all, Associative Design Framework including OOD, AAD, Procedural Modelling and relationships between them are proposed as a basis for Algorithm-aided Information Modelling methodology.



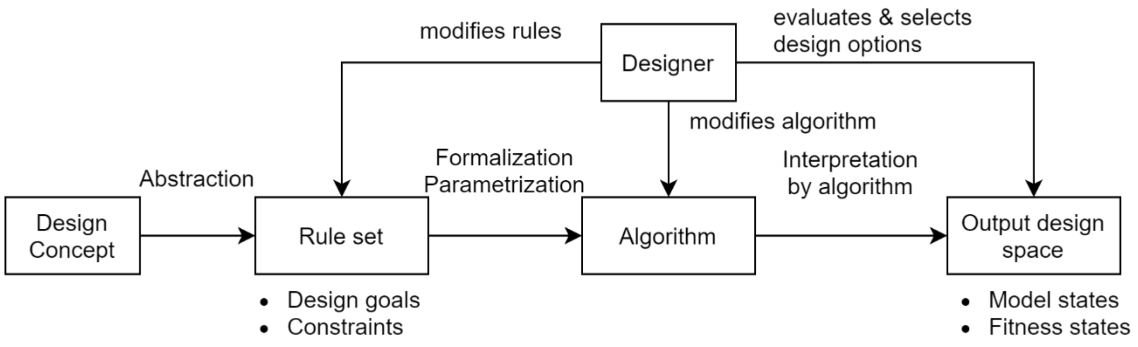


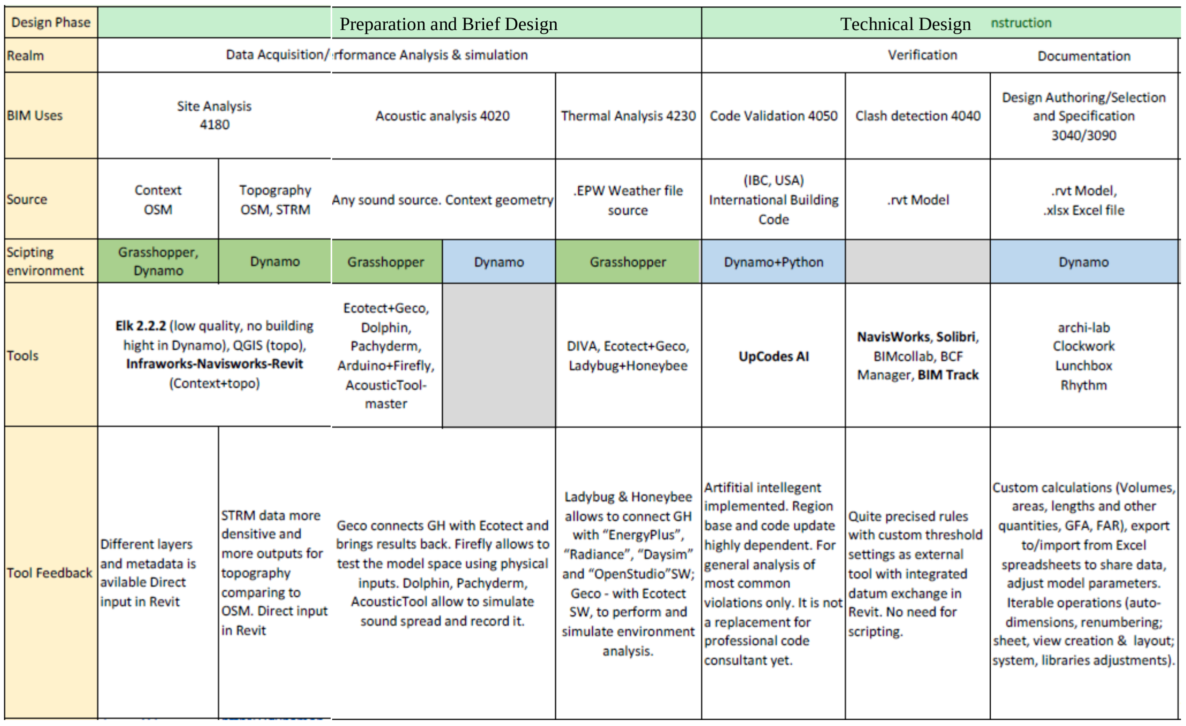
Figure 1.7 – Projects with applied Generative Design in architecture, urbanism, and in object design (top row from the left to right): Columns cut with CNC and assebled by Michael Hansmeyer; Kartal Masterplan by ZHA, Istanbul; Moebius ring by Torolf; In the bottom: Generative System Workflow based on Hartmun Bohnacker, book "Generative Gestaltung"

1.5. RESEARCH METHODOLOGY

The present research primary type is Exploratory Research with mixed complimentary research typology criteria. Specifically, except literature review and exploration of new phenomena of Algorithm-aided Information Modelling, it gains the explanation of relationships and correlations between concepts such as OOD and AAD. Meanwhile, certain design problems are solved, analyzed, and reported in Practice part within Design Cases provided by partner company "NOZ Arquitectura" in Lisbon, Portugal. Hence, Case Studies Analysis with sufficient reports and process descriptions points out the Applied Research type. Besides, quantitative technical information specific to AEC and IT industries, and qualitative information, including textual description complemented with graphics reveals its Mixed Research nature overall.

1.5.1. RESEARCH FRAMEWORK

The Research Framework proposed in the MSc dissertation is presented in Figure 1.9. It contains three main stages consequentially: Theory Basics: Conceptual Framework, Practice, and AID Theoretical Synthesis. Theory Basics is written in Chapter 1 Introduction. It includes an establishment of the subject of research, the definition of objectives for the study, and literature review of the most relevant material as well as a survey of existing tools and workflows within the State of Art. In the end, the AIM Conceptual Model and Framework are proposed. Literature Review covers the brief history of CAD, the transition from Additive to Associative Design embracing OOD and AAD methodologies, their advantages and drawbacks. Next, State of Art is explored, which resulted in the table in Figure 1.8. In the table, existing workflows and utilized AIM tools are mapped throughout the design process by BIM Uses (BIMe, 2020) on different design stages according to RIBA Plan of Work (Anon., 2020). The table also contains Source of data information as well as GAE, and Tool Feedback. Further in Theory Basics, Conceptual framework for digital design is defined to determine meanings of main concepts and relate them with each other. AIM Conceptual Model is presented as the basis for further investigation of methodology.



Design Phase			Preparation and Brief Design			Technical Design	nstruction	
Realm	Data Acquisition/		rformance Analysis & simulation				Verification	Documentation
BIM Uses	Site Analysis 4180		Acoustic analysis 4020		Thermal Analysis 4230	Code Validation 4050	Clash detection 4040	Design Authoring/Selection and Specification 3040/3090
Source	Context OSM	Topography OSM, STRM	Any sound source. Context geometry		.EPW Weather file source	(IBC, USA) International Building Code	.rvt Model	.rvt Model, .xlsx Excel file
Scipting environment	Grasshopper, Dynamo	Dynamo	Grasshopper	Dynamo	Grasshopper	Dynamo+Python		Dynamo
Tools	**Elk 2.2.2** (low quality, no building hight in Dynamo), QGIS (topo), **Infraworks-Navisworks-Revit** (Context+topo)		Ecotect+Geco, Dolphin, Pachyderm, Arduino+Firefly, AcousticTool-master		DIVA, Ecotect+Geco, Ladybug+Honeybee	**UpCodes AI**	**NavisWorks**, **Solibri**, BIMcollab, BCF Manager, **BIM Track**	archi-lab Clockwork Lunchbox Rhythm
Tool Feedback	Different layers and metadata is avilable Direct input in Revit	STRM data more densitive and more outputs for topography comparing to OSM. Direct input in Revit	Geco connects GH with Ecotect and brings results back. Firefly allows to test the model space using physical inputs. Dolphin, Pachyderm, AcousticTool allow to simulate sound spread and record it.		Ladybug & Honeybee allows to connect GH with "EnergyPlus", "Radiance", "Daysim" and "OpenStudio"SW; Geco - with Ecotect SW, to perform and simulate environment analysis.	Artifitial intellegent implemented. Region base and code update highly dependent. For general analysis of most common violations only. It is not a replacement for professional code consultant yet.	Quite precised rules with custom threshold settings as external tool with integrated datum exchange in Revit. No need for scripting.	Custom calculations (Volumes, areas, lengths and other quantities, GFA, FAR), export to/import from Excel spreadsheets to share data, adjust model parameters. Iterable operations (auto-dimensions, renumbering; sheet, view creation & layout; system, libraries adjustments).

Figure 1.8–Table "State of Art on workflows and tools of AID"

Practice stage is dedicated to the investigation of the subject through experimentation and real design problems solving within Case Studies. It is divided into four Case Studies, respectively, embracing prominent aspects of the design process and covering Brief Design, Technical Design, and Manufacturing Stages according to RIBA Plan of Works (Anon., 2020). These aspects include Site Data acquisition and analysis, Conceptual Design Development, Interoperability, Documentation authoring, and Manufacturing. More specifically, in the Case Study 01, Context Model is created using AIM tools, and Site Analysis is performed. In Case Study 02, Views Generation and Automatic Layout on a sheet are considered. Next, Case Study 03 examines design-to-fabrication workflow where Panelization tool is developed for transferring manufacturing data from the BIM system to a CNC machine. Finally, The Case Study 04 concludes with AIM system development and testing it within workflow combining different environments and tools.

After Practice Part of the research, Conceptual Framework Analysis is performed compared with the framework from Theory Basics part. Next, endeavor for its improvement is undertaken, and the dissertation research is concluded in AID Theoretical Synthesis.

Nonetheless, stages are not directly sequential as they all interconnected with bi-directional requests for reports. Regarding the Research Framework peculiarities as intrinsic to evolution project management needs, the Agile methodology is applied for building framework with its iterative and incremental methods. Consequently, the framework is considered as incrementally cyclic because Theory and Practice parts are revisable during all the stages of research. Particularly, during Case Studies stage, it is necessary to revise related literature extensively through the lens of the specific case study. Besides, rethinking of Conceptual Framework in AID Theoretical Synthesis requires theory review from Theory Basics as well as Practice analysis reports. Therefore, the research timeline is intentionally left formal and generalized over real-time spans for feasible representation. (Figure 1.9)

Ultimately, research methodology is based on a playful approach which was established by Alvar Aalto in 1953. He stated that technology and economy should "always be linked to the charm, which makes life richer." In Aalto's endeavour, standardized technologies are challenged with creative customization. He said: "Flowers of the apple trees are standardized, but they are all different. Thus should we learn to build". The same approach is seen appropriate in the present research to determine and apply an integrated methodology of Algorithm-Aided Information Modelling, which should increase playfulness holding on complexity and accuracy at the same time. All in all, the Agile methodology and Playful approach are synthesized in this study methodology and Research Framework.

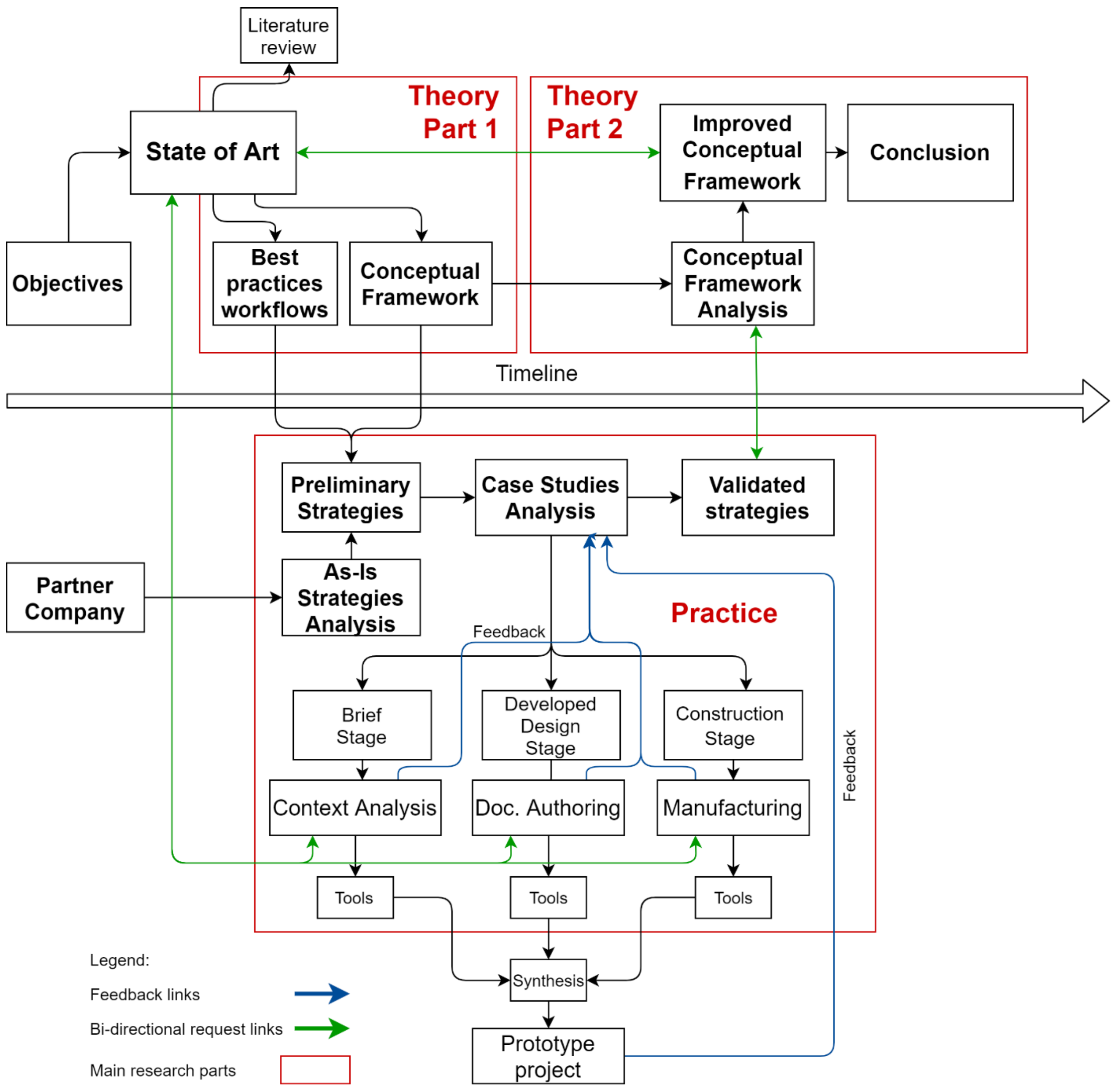


Figure 1.9 – Research Framework Chart

This page is intentionally left blank

2. CASE 01. SITE DATA ACQUISITION AND ANALYSIS

2.1. INTRODUCTION

Site analysis is one of the very first steps in the architectural design process. When starting a new project, the first thing to look for is usually a fair representation of the project's environment.

It is important to inform the design strategy and to justify a proposal with the client and planning authorities. Detailed knowledge of the context will contribute to the establishment of the proposal on the site and help justify the proposed design. There are many ways to do so already, but they all require using external sources and export/import of information before having them in the authoring tool. Context information can originate from a local survey, site photos, point clouds, and online services such as Open Street Map, Google Earth, USGS and others. However, these are usually very focused on the intervention site and miss the wider context. Furthermore, the information is scattered and not synthesized on a unique model. Additionally, a project may be in a distant location, and financial commitment may not allow for certain tasks to be carried at early project stages. To populate a model, the context has to be manually modelled based on the above assets, which is time-consuming and not very valued. (Dias, 2020) Therefore, the challenge is to extract, organize, and operate the data from a wide range of sources in a smart and automated way. To achieve such a goal, computational tools with scripting power can be implemented.

Site analysis is widely used on early stages of design by a variety of means and objectives. It is a comprehensive definition that covers different aspects:

- Site Geometry data including toposurface, buildings, roads, public spaces, etc
- Site GIS data including transportation system, amenities, facilities, and services
- Site Climatic & weather data

All of these datums are used on different stages during the whole project lifecycle. Within the BIM industry, there are defined Model Uses matching the above aspects with specific codes (BIMe, 2020). Model Uses identify and collate the information requirements that need to be delivered as embedded within – 3D digital models.

Consequently, Site Geometry is represented by Laser Scanning (2050), Photogrammetry (2060), Landscape Modelling (1240), Terrain Modelling (1440), Urban Modelling (1490), Clash Detection and avoidance (4040).

Site GIS Data covered by BIM/GIS Overlapping (8040) Infrastructure Systems Modelling (1220), Traffic Modelling (1460), Transportation Systems Modelling (1470), Site Analysis (4180), Risk and Hazard Assessment (4150), Accessibility Analysis (4120), Egress and Ingress (circulation) Analysis (4080), and Acoustic Analysis (4020).

Site Climatic & weather data is used in the Lightning analysis (4010), Solar Analysis (4190), Thermal Analysis (4230), Energy Utilisation (4090), Wind Studies (4260), Sustainability Analysis (4220), and others.

As seen above, quite a lot of Model Uses require Site Data considering different aspects. Therefore, as a complex information asset itself, the site carries huge and versatile datum which is necessary to deploy by inquiry according to specific needs, retaining efficiency.

OBJECTIVES

To meet information requirements according to Model Uses and to make delivery efficient, it is important to clarify goals before site modelling. Consequently, the next goals are identified:

- Building Performance analysis and simulations
- Clash Detection
- Presentation

Depending on Site modelling goals, it is recommended to keep a certain level of development (Anon., 2019) and specifically optimized workflow. For instance, Clash Detection requires solid geometry of surroundings. Meanwhile, to simulate and analyze Building Performance, climatic & weather data is acquired along with geometry. In comparison, point clouds (converted to mesh), orthophotoplans, and photoshoots most suit for Presentation purpose. The information source is vital for Site acquisition and modelling and may cause unreliability as well as overloading model with waste data.

Thus, the main objective is to develop the appropriate Site Model, which will meet all the goals specified by the partner company, and examine it for usability afterwards. The general approach is reflected in the framework in Figure 2.1. The intent is to synthesize all workflows related to site keeping optimization and focus according to LoD and specific needs on different stages and of parties of the project. The output should be a parametric Site Model in Revit with the ability to extract and analyze GIS data as well as to visualize context. The model should be linked with the point cloud. The point cloud is assumed to be an adjustable asset with control area of coverage and overrides for presentation purposes. Each asset is regarded as a parametric family instance with parameters enriched by GIS data which afterwards can be analyzed through scripts under different possible simulations directly from Revit. Finally, analysis & simulation scripts should be developed to test the model.

As a location for Site Model, São José Building and Jardim Visconde da Luz Garden in Cascais were proposed by the partner company for the investigation. Three execution projects are currently work in progress there. One of them is the garden place - Visconde da Luz Garden. It is intended to renovate the spot with a modern design to include an open-air amphitheatre, to allow music plays to return to the garden as it was once a tradition, to accommodate the carousel better and create a more inviting for people to stay and enjoy the public space. Another part of this exercise, NOZ Arquitectura proposed for the adjacent São José Building to be partially renovated so that it would create a new back ground to the new park, and saw an opportunity to enhance the connection to the nearby market.

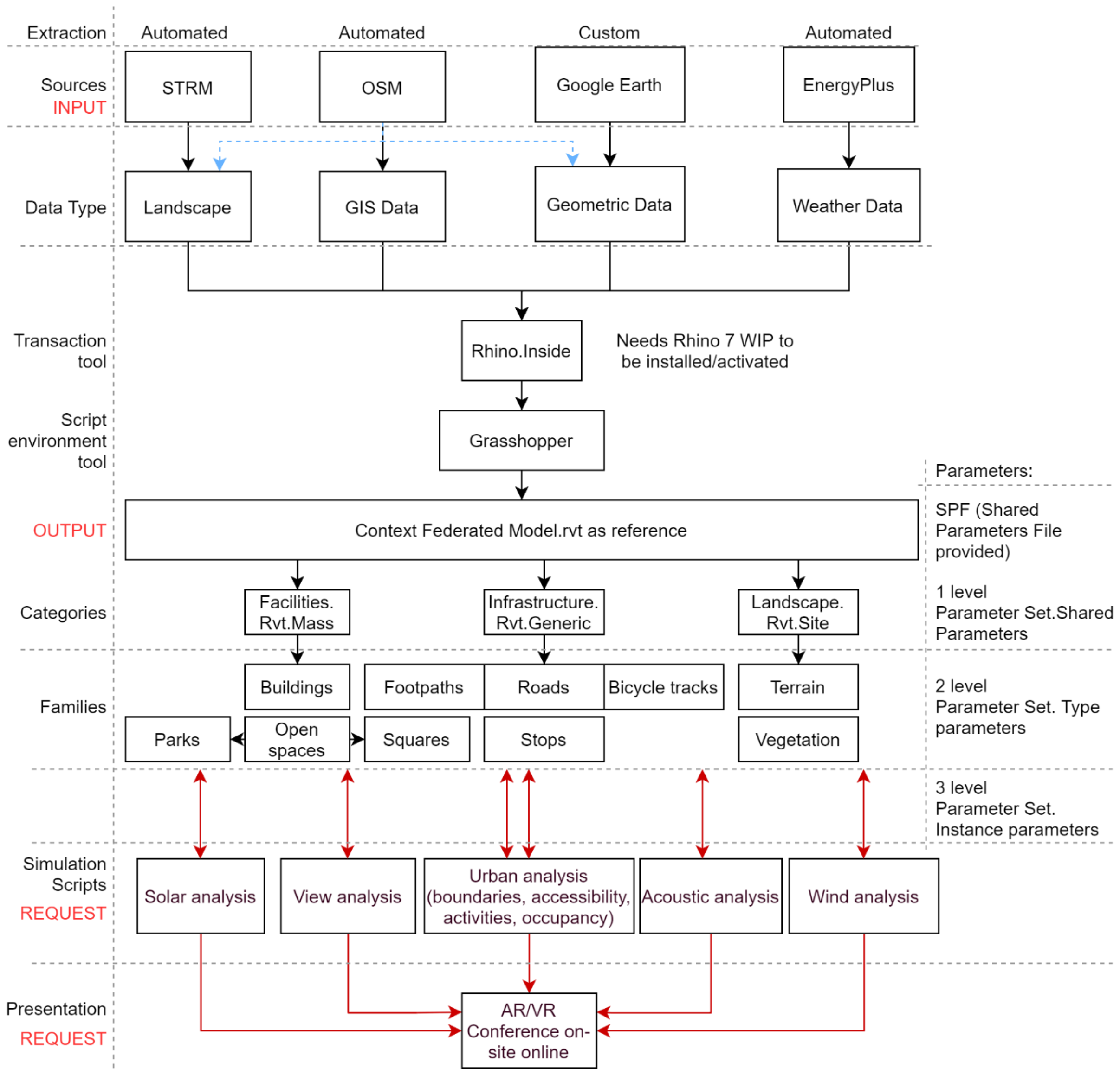


Figure 2.1 - Site Model Conceptual Framework

To build the argument for a wider intervention, the practice modelled the context to various levels of detail to produce its diagrams and convincing visuals. The current case builds on the requirement for the architect to sometimes look into the wider area of intervention and find a workflow to quickly create context models that can be used for analysis, the definition of architecture strategies and presentations.

CONTEXT ANALYSIS.

To perform the context analysis, GIS data is the most useful, as we consider context as a group of entities with common attributes that can be compared and analyzed consequently. Plenty of tools have been elaborated for GIS data acquisition such as Elk, Flux, GIS2BIM, DynaMaps, and Infraworks. All of them based on Open Street Map as a source database. Though means of extraction and interpreting information is different, still the majority of them utilize XML format for converting and packing data. Some tools are cloud-based (Flux, DynaMaps, Infraworks). Although Flux is unavailable anymore, it is a great example of the cloud-based platform could work broadly and effectively for collaboration and analysis in the AEC industry. Meanwhile, others are local interpreters with the need to export/import GIS data manually (Elk, GIS2BIM) At the same time, they usually have more functionalities. The lack of some tools (GIS2BIM) also is region localization since many professionals elaborate tools under their needs within-country regulations and databases. Lastly, Infraworks is a stand-alone software for site design, analysis, simulations and visualizing the concept design in a real-world context. The reason why it is also considered in this case is that Infraworks is closely integrated with BIM authoring tools such as Revit as part of Autodesk software family. Therefore, it is comparable with other scripting tools in terms of delivery Site Model in Revit.

As a result of the literature review, two main tools were chosen for the context analysis task: Elk and DynaMaps.

Elk is a set of tools to generate maps and topographical surfaces using open source data from OpenStreetMap.org and USGS. Elk organizes and constructs collections of point and tag data enabling the creation of curves and other Grasshopper/Dynamo geometry.

Elk allows us to build topography in Revit and map GIS context on it. The weakness is that it is not equally spread data all around the world. So, for example, many buildings don't contain information about its height. Thus usual workflow assumes a workaround with randomizing height within a known range which is not a reliable way for transferring GIS to BIM. However, it is still appropriate for some representative purposes and preliminary analysis.

Next tool for implementation is DynaMaps. This is a project originated at the London Hackathon of April 2019 and is now available on the Dynamo package manager. The project is dedicated to the geometrical representation of the project's environment on the preliminary design stage. The tool reveals a straightforward workflow fully accommodated within Dynamo GAE, unlike most other analogies using external sources and export/import of information before having them in the authoring software. It is an attempt to bring back Flux.io experience and develop it even further to get site data into Dynamo.

DynaMaps provides more detailed geometry, including topography, buildings, roads, and trees. It automatically projects geometry on toposurface without need to extrude footprints manually with splitting solids by toposurface afterwards. Besides, DynaMaps provides more accurate roadways with original width, unlike Elk provides just polylines. Comparing to Elk, DynaMaps is more automated and detailed but gives less control manipulating data acquired from the same source Open Street Map. So, it does not contain GIS metadata such as building address, levels, names, etc. It is also notable that to generate buildings, node ProjectBuildingsOnTopo requires Buildings Elevations which can be acquired only from Building Foot Print Surfaces. The problem is that FootPrintSurfaces node is the most hardware performance demanding to compare with the rest of the package components. To optimize the workflow, it would be better to filter FootPrintPolygons first, which is much less demanding, by distance constrain. Then we could patch polygons into surfaces for ProjectBuildingsOnTopo input. In this way, we would save plenty of computational time and avoid errors.

2.2. CASE STUDY. SITE MODEL

The workflow consists of three stages and three scripts accordingly. Initially, a script for the topography generation is performed; a second script is used for mapping entities such as buildings, roads, walkways onto toposurface and then generate its geometry within user-defined distance range from the project spot. Finally, a third script is elaborated for site analysis on the base of created Site model.

2.2.1. 1st Milestone:Topography Generation

The first script is dedicated to topography generation. Three methods were taken into consideration according to each tool. Elk tool requires OSM file for location input and geoTiff file for topography points extraction. First, the .osm file is downloaded from Open Street Map WebSite (Anon., 2020). Then .tiff file is downloaded as well from USGS (United States Geological Survey WebSite, (Anon., 2020)). (Figure 2.2) Further both files are connected to the definition through File Path Node. The Node Topography.CreateTopo provides points, curves, and surface geometry in DesignScript environment. To translate this geometry to Revit native topography element, list of outputs points is flattened and connected to Topography.ByPoints Node which finally bakes the geometry in Revit. (Figure 2.3)

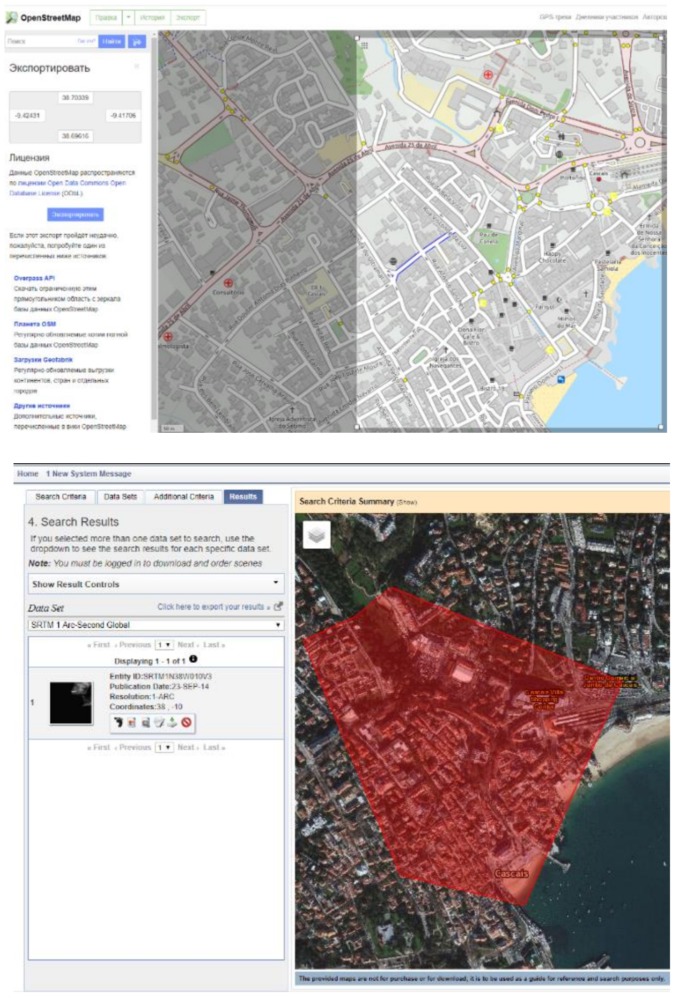


Figure 2.2 - OSM file export (on top), GeoTIFF file export (on bottom)

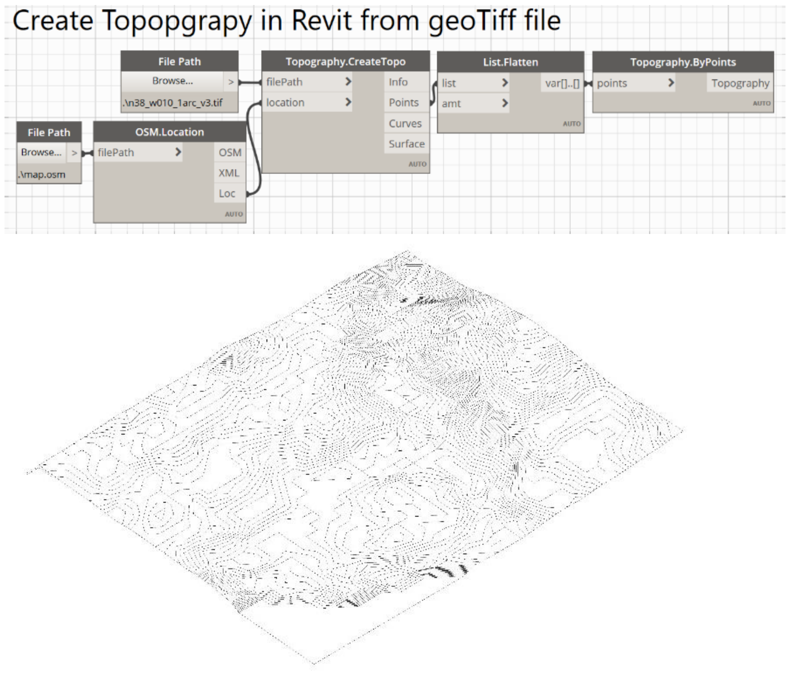
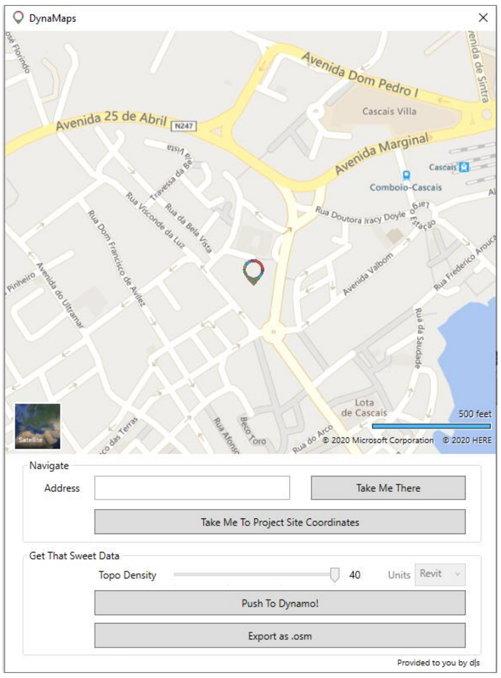


Figure 2.3 - Topography created by Elk node

Alternatively, DynaMaps allows gathering GIS data straightforward from the OSM database through its special view extension, which can be found under the “view” tab of Dynamo. (Figure 2.4) The map viewer looks like map online service. It uses Bing Maps as a reference. To extract OSM data, it is required just to zoom and navigate with the mouse, or use the “Address” textbox and click “Take me there,” to navigate directly to the location. DynaMaps uses the Nominatim API for geocoding. Thus it is important to be as precise as possible giving the address. After selecting the place, it is needed to press the button “Push to Dynamo”. Notably, there is also a feature to export .osm file. This feature allows to combine the use of DynaMaps with other OSM tools, which will be implemented further. The view extension fetches the OpenStreetMap data corresponding to the area shown on the map, and then the DynaMaps node TopoPoints is used to output points and bake the topography through Topography.ByPoints the same way as Elk tool. Remarkably DynaMaps allows choosing the toposurface density in the range [10;40] of points in UV directions which should increase the quality of toposurface. However, after close investigation and comparison between different density settings in DynaMaps and Elk Topography, it appeared that DynaMaps increases density by interpolating point coordinates which are estimated among the established set of other points. Consequently, topography area does not matter. Toposurface of any size is calculated on the base of input density value in UV directions. Unlike DynaMaps, Elk provides a fixed grid of points. So, the less area is selected, the less amount of points is calculated. From our point of view, both methods can be justified. Elk provides just initial and correct coordinates data from OSM. Meanwhile, DynaMaps populates density by interpolating values. Point density comparison is represented within the range available in DynaMaps and toposurface generated by Elk in Figure 2.8. As a result, DynaMaps tool is chosen with density value 40 for topography generation. (Figure 2.5) The definition is very simple and consists of literally two main nodes Topo.Points and Topography.ByPoints and some additional nodes to preview the GIS data in Dynamo viewport such as roads, walkways, and buildings. (Figures 2.6, 2.7)



Topo Density 40
Units Revit

Figure 2.4–View extension DynaMaps

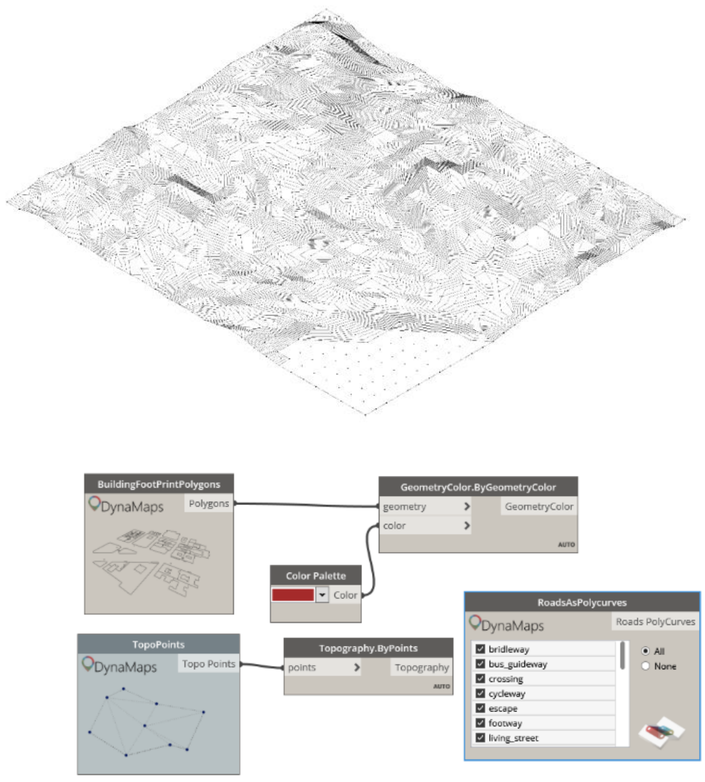


Figure 2.5 - Topography By DynaMaps. Density 40

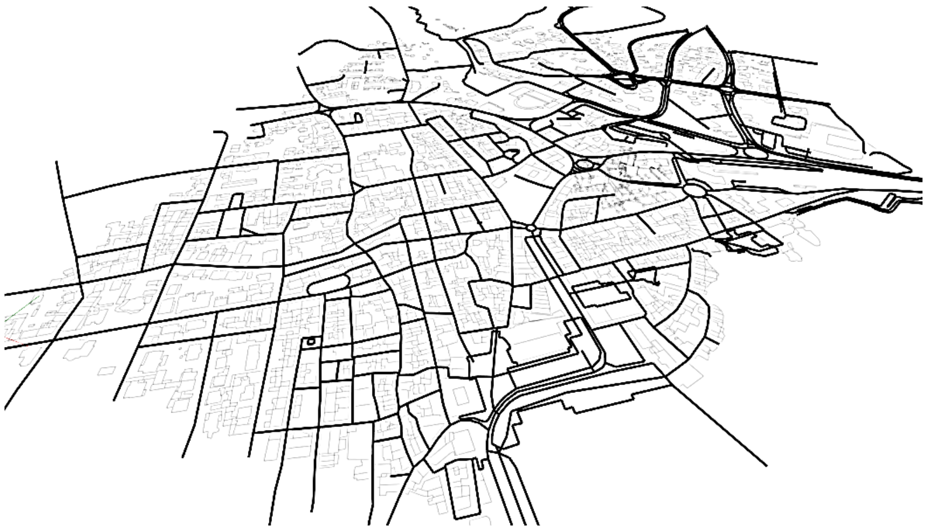


Figure 2.6 - Roads. Elk node OSM.OSMData. Dynamo

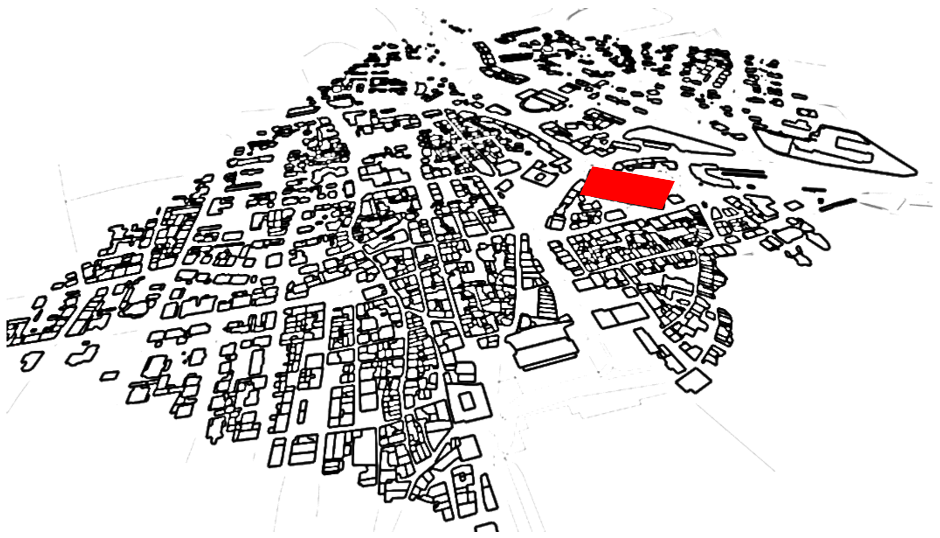


Figure 2.7–Buildings. Elk node. OSM.OSMData. Spot in red, Dynamo

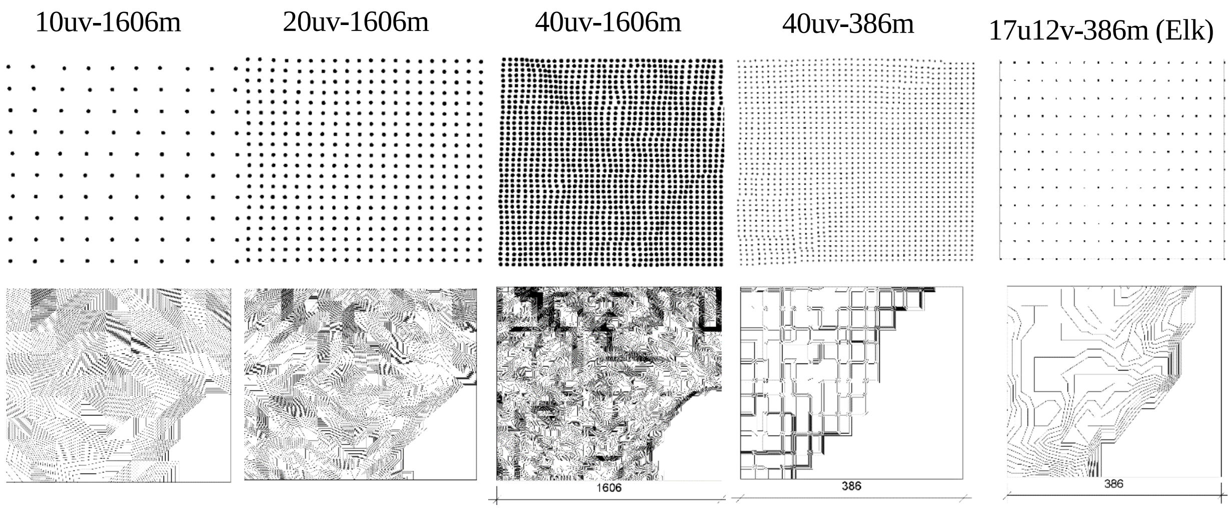


Figure 2.8 - Topo density comparison diagram: DynaMaps vs Elk (on right)

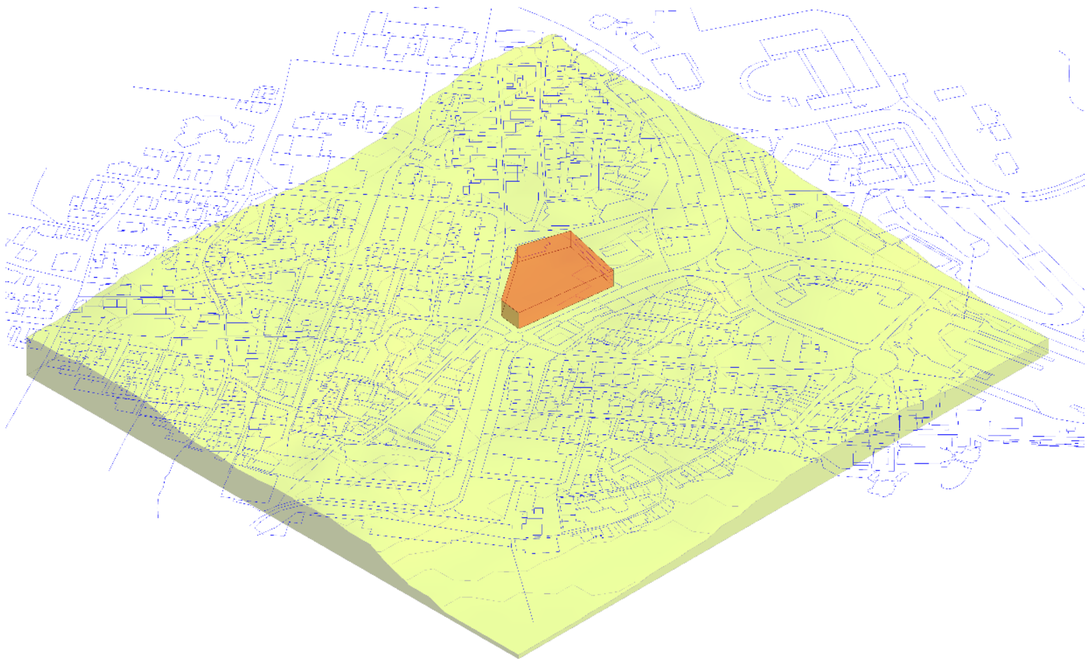
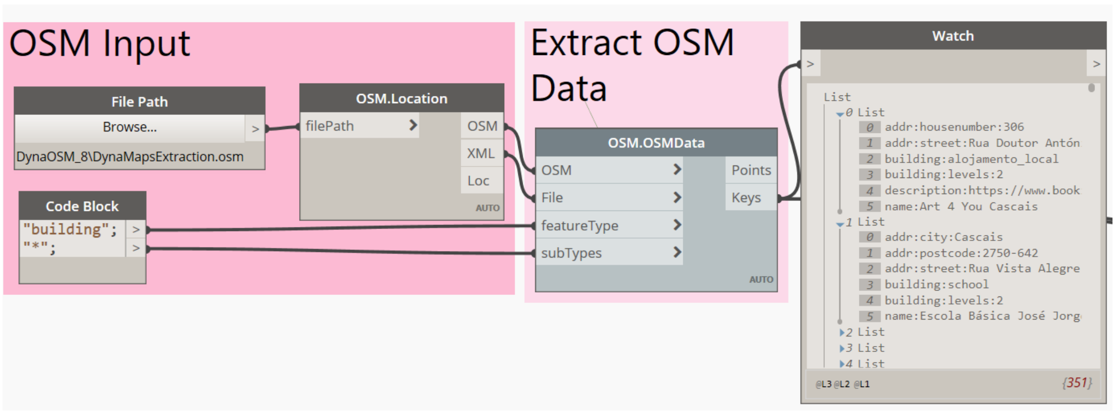


Figure 2.9 – Result of the topography generation after the first script running

2.2.2. 2nd Milestone: Convert GIS Datasets to Revit Components

After Topography is created, spot boundaries should be drawn on the base of topography and preview GIS geometry generated in the first script. (Figure 2.9) Boundaries can be made by detail lines (closed-loop), by region or by mass. The goal of the second script is to bake GIS geometry as families within constrained diameter from spot centre and to attach parameters for further analysis.

Elk tool has just single node OSM.OSMData for extraction any data as point dictionary with key metadata values. It requires location and feature type and subtype for inputs. Metadata attached can be address, building type, levels, and name for building feature type. Remarkably, metadata set varies by entities within the same feature type depending on available GIS data from OSM. For instance, in Figure 2.10 in the Watch node, it is seen that for the second building in the list address content is compared with the first one. It contains postcode and city attributes. Meanwhile, the house number is absent. It is important to take in consideration that some entities will have empty common attributes at the analysis stage.



```
List
  0 List
    0 addr:housenumber:306
    1 addr:street:Rua Doutor Antón
    2 building:alojamento_local
    3 building:levels:2
    4 description:https://www.book
    5 name:Art 4 You Cascais
  1 List
    0 addr:city:Cascais
    1 addr:postcode:2750-642
    2 addr:street:Rua Vista Alegre
    3 building:school
    4 building:levels:2
    5 name:Escola Básica José Jorg
  2 List
  3 List
  4 List
```

Figure 2.10 – Elk OSM data extraction in Dynamo

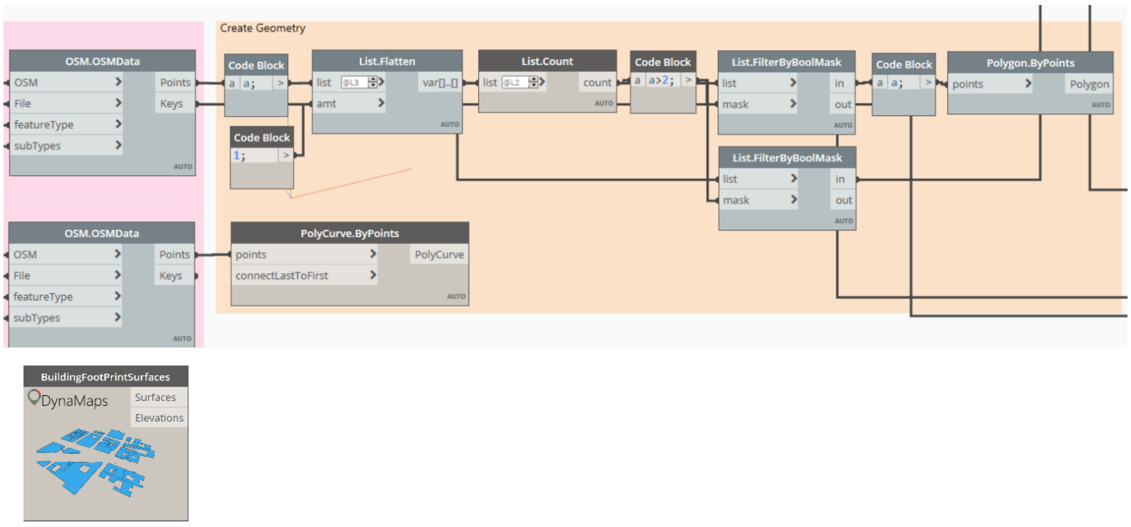


Figure 2.11–Elk vs DynaMaps: Create building footprints from OSM data (Dynamo)

Next, the contour polygons are created on the base of the extracted points. Some point sets consists of two points and provide an error in creating polygons. Therefore filter is applied to remove such sets. (Figure 2.10) Further, polygons can be projected to topography created in the first script, filtered and extruded as solids. However, Elk does not have elevation values for most of the buildings in the case area. It is still a consequence of uneven OSM coverage worldwide. The workaround could be to set random values for building elevations, but it was not intentional.

Therefore, the decision was to use DynaMaps again due to its more accurate elevations data and interpolation approach for the intervening values. Though, it is still far off the mark. To create building footprints by DynaMaps, it is enough to insert single node BuildingFootPrintSurfaces which represent footprints itself and contains elevation data attached. (Figure 2.11) Once we have elevation data and footprint geometry, it is time to filter context by distance constrain. Distance is defined by user input boundary for the project spot. It is drawn by detail line (close-loop), region, or by mass on the base of topography created in the first script. Then boundary is selected by node Select Model Elements and centre of the polygon is found. Next, the distance calculated from the centre point to footprints. Further, distance is compared with the user-defined input value. If the distance is more then the input value, then footprint is filtered. Elevations filtered by the same rule. By this way, footprints of the context are gathered and ready to be extruded. Another aspect is that we need to define toposurface to extrude from. To do so, the topography is selected from Revit viewport and converted to polysurface for allowing projection of footprints. Next, filtered footprints are projected on the converted toposurface and extruded by elevation value. To raise building geometry DynaMaps has two nodes. One is for Revit families generation, another one for dynamo geometry generation. Both nodes are straightforward and require footprints and elevations for input. However, the first node is very heavy in terms of computation and takes usually a lot of time freezing system, thus providing the risk of losing information. Therefore, first, dynamo geometry is generated which is next intended to be baked as family instances in Revit. (Figure 2.12) Besides this interim step allows testing generated context in Dynamo viewport before sending it to Revit. For this purpose, solids and toposurface are visualized accordingly.

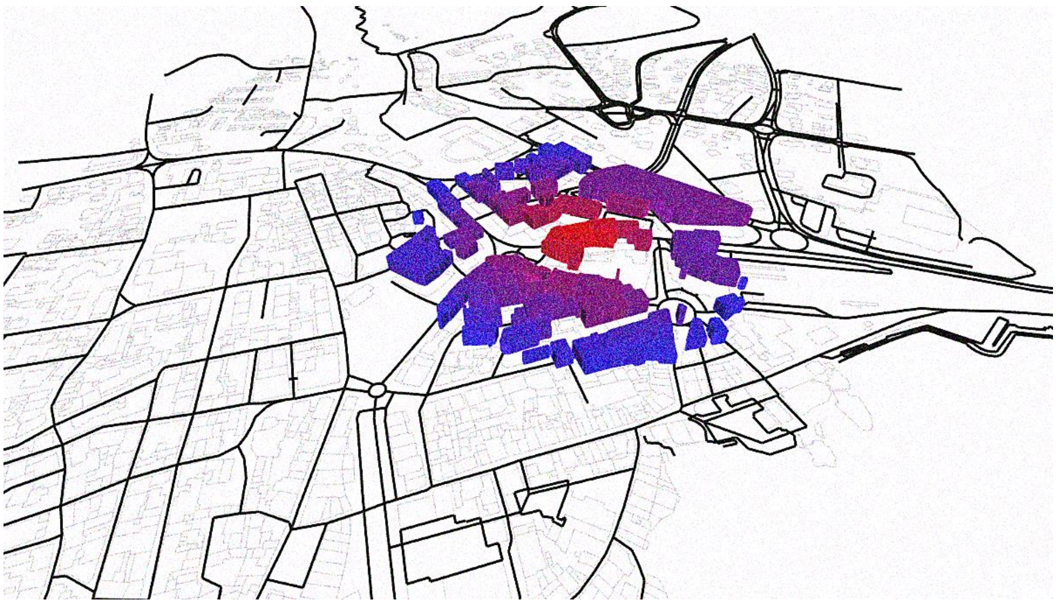


Figure 2.12 – Constrained & filtered context in Dynamo Viewport by distance to spot

After geometry is generated, the further step is transferring metadata attached to each family instance. Since DynaMaps (DM) does not have metadata extracting feature, Elk tool comes to aid again at this point. Further script development derives into two optional workflows based on the combination of tools. (Figure 2.14) The first workflow includes footprint points and metadata using Elk tools and only elevations from DynaMaps. Such workflow is aimed to avoid overloading system not involving heavy DM node BuildingFootPrintSurfaces in generating operation. Instead just utilizing extract elevations from DM is assumed. Nevertheless, the weakness of such an approach is the need to align coordination systems between two tools involved. As a result, we have to utilize both tools and align topography from DynaMaps with footprints from Elk as it appeared to use different coordination systems initially. (Figure 2.13)

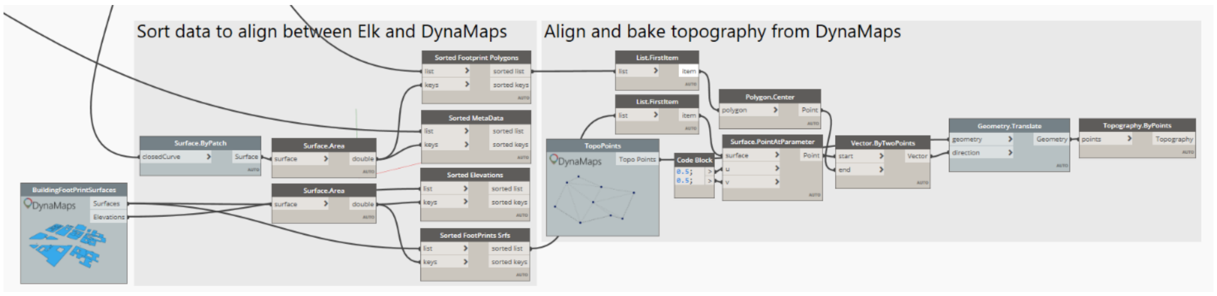


Figure 2.13 - Sorting and alignment data definition part between Elk and DynaMaps tools

Alternatively, according to the second workflow, all the data, including topography, elevations, and footprints are gathered from DM. Then Elk metadata is sorted by footprints and data sets are aligned to geometry entities. (Figure 2.14, on the right) By this way, we avoid alignment of coordination systems.

Finally, Site Model is created constrained by distance range from project spot and with attached GIS metadata such as address, building type, name, and levels. This type of Site model fully gathered from OSM GIS database. It well suites to the preliminary site analysis. Though it has some lack of information depending on available data of site location. Geometrical lack mainly solved by interpolating by DynaMaps tools. Meanwhile, Elk provides metadata about GIS objects which further can be added as well.

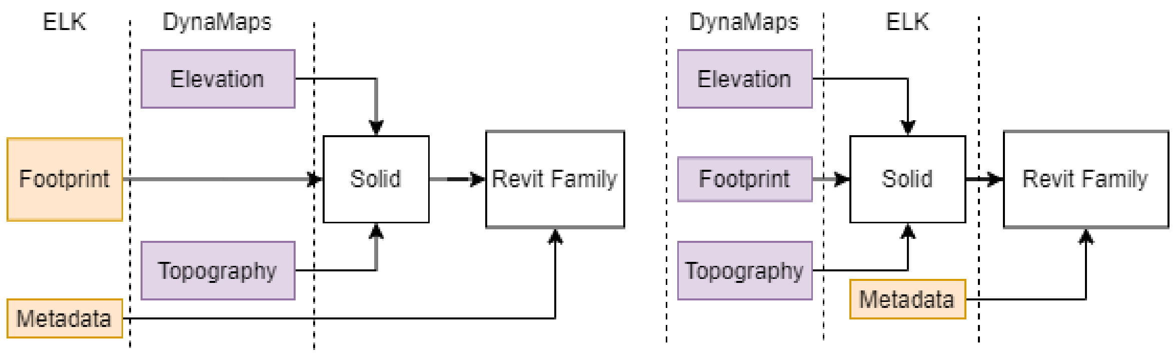
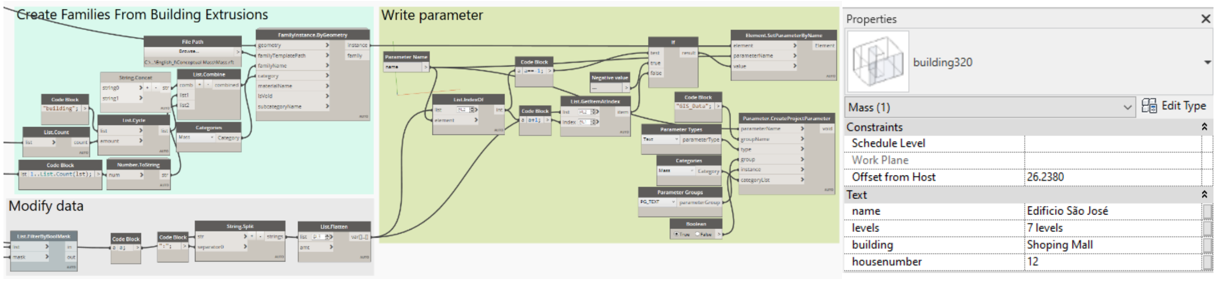


Figure 2.14–comparison scheme between two workflows creating parametric family from GIS



Properties

building320

Mass (1) Edit Type

Constraints	
Schedule Level	
Work Plane	
Offset from Host	26.2380
Text	
name	Edificio São José
levels	7 levels
building	Shoping Mall
housenumber	12

Figure 2.15–Transferring parameters from OSM to mass family instance

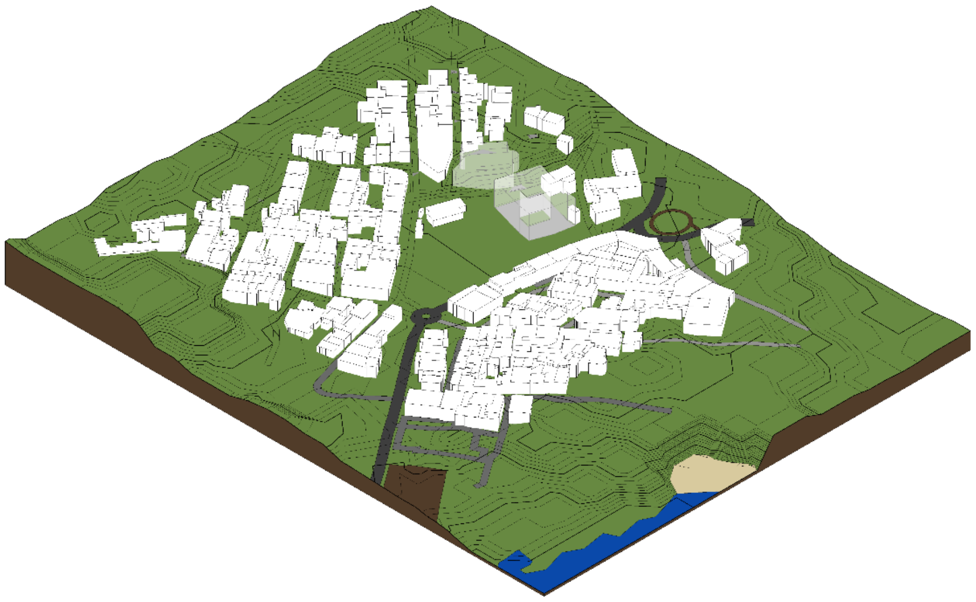


Figure 2.16 – Result Site Model after second script running

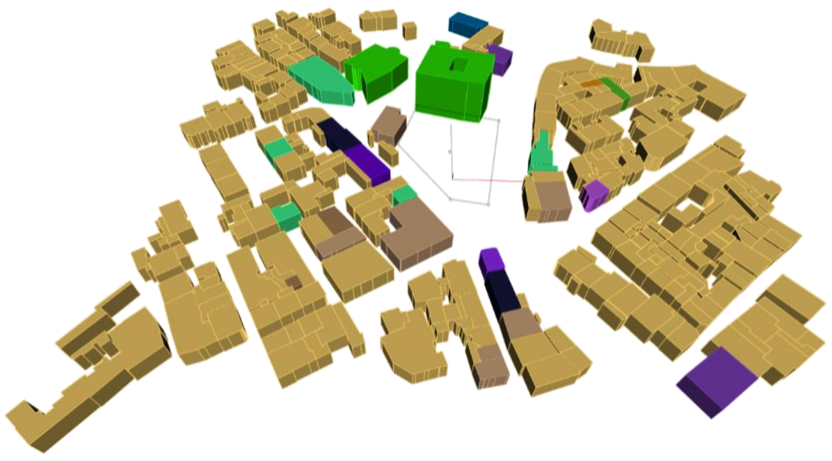
2.2.3. 3rd Milestone: Site Analysis.

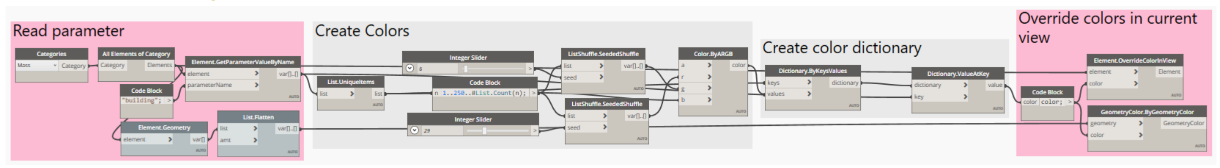
Next part of the case study is Site Analysis. Site Analysis includes building performance studies such as Lightning and Thermal Analysis as well as Infrastructure analysis, including context Facility Typology, Accessibility Analysis, etc. Infrastructure data is gathered from family instance parameters transferred from OSM prior in the second script. Unlike infrastructure data, most of the building performance analyses require climatic and weather data. Such data can be extracted through dedicated packages Honeybee and Ladybug.

All of the infrastructure analyses are scripted similarly. The definition consists of four sequential parts:

1. Read parameters of family instances
2. Interpret parameter values
3. Create a colour dictionary with interpreted value keys
4. Override graphics in current view

A typical instance of infrastructure analysis is facility usage in the context area. To demonstrate usage analysis, it is needed to extract unique values from parameter “building” which was generated during the second script and assign colours according to each type of usage. Since the number of typologies is variable within context, generated colours are controlled by the input count of unique keys of extracted typologies. (Figure 2.17)





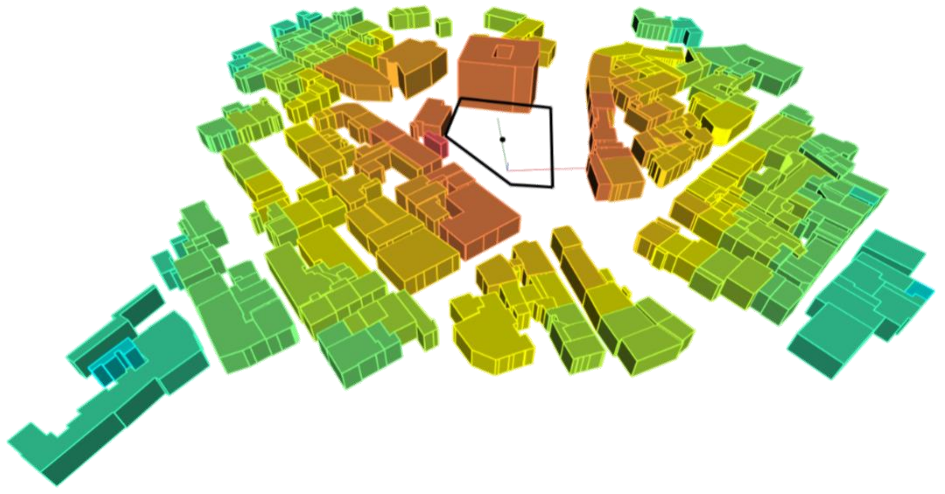
```
"building";
```

```
n 1..250..#List.Count(n);
```

```
color;
```

Figure 2.17 - Infrastructure analysis. Dynamo viewport (top), script definition (bottom)

Another example of infrastructure analysis is Accessibility Analysis. The analysis is based on the distance range inclusion test. Input of definition is selected spot boundary by detail lines or region. Next, the central point of spot is determined and distance to context buildings geometry is calculated by node Geometry.DistanceTo. Once we have list of calculated values, we need to remap it into the range [0;1] to be able to input values for colourization, but before remapping procedure one more step is round distance values by a user-defined number. So, buildings and their colours can be grouped by round values. Eventually, all the round values are remapped by node Math.RemapRange and colour range created by these interpreted values within range [0;1]. The next part of definition is creating colour dictionary with distance key values. Consequently, colours are assigned to building elements according to their distance values to spot centre and visualized both in Revit current view, and Dynamo view. (Figure 2.18)



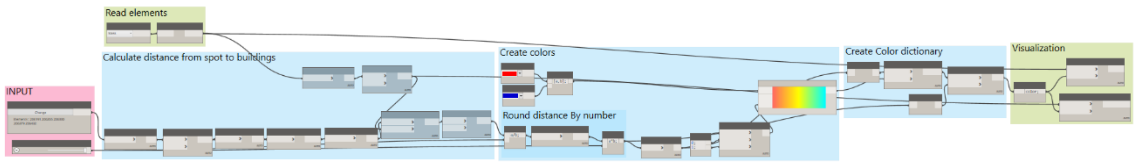


Figure 2.18 - Accessibility Analysis definition (bottom), and its visualization in Dynamo viewport (top)

2.2.4. Point Cloud Integration

A point cloud is a fast and effective solution for presentation and model quality assurance tasks. However, the point cloud still requires on-site survey with special and expensive equipment such as laser scanner and drone. Survey service can cost considerable money. To spend company means to collect context data on-site before the proposal is excepted is inappropriate. Therefore, a new approach was investigated. The objective was to find a way to extract point cloud online and integrate into Site Model.

After an initial research and literature review, the complete online method was revealed without the need of the physical survey of the site. This method applies photogrammetry as technology. Photogrammetry is the process by which two-dimensional images are translated into three-dimensional measurements or models. The process involves analyzing and comparing photos taken from the air or terrain. Each photo typically offers a different field of view. Fields of view are overlapped to calculate measurements. To increase the quality of the model, several reference points (Tie points, GCPs) on the target that are visible in the available photos are used as well. (Goodman, 2020) Using GCPs, different camera angles, and fields of view allow us to triangulate the position of the points within the 3D space. (Figur 2.19) The output can be point cloud, orthophotoplans, maps, or measurements (distance, area, volume). Implementing photogrammetry, Andre Agi (2019) has created a workflow gathering point cloud from Google Maps. Script hat makes screenshots with defined intervals while navigating on Google Maps was the basis of the workflow. These screenshots further proceed in Adobe Photoshop to calculate a mesh or point cloud from aligned and overlapped images. The more screenshots are made, and the more fields of view are covered within the same target objects, the more will be the quality of output.

In the case study, Andre’s workflow was adopted by intent to simplify the workflow making it available on any PC without even running the script. So, video screen record can be made by any default application on PC or Mac. In this case study, Xbox Game Bar was used for this purpose. After flying over the 3d target area in Google Earth (Figure 2.20), the screen video proceed in Adobe Premiere Pro where frames are extracted. It is possible to control the number of images and overlap by simply adjusting clip speed in Premiere Pro. (Figure 2.21) Consequently, the number of frames and overlap effects on quality and alignment speed during point cloud generation. Alternatively, it can be done with default video editors as well. Next, extracted frames can be edited in any photo editor for colour correction and cropping. To achieve a better quality of point cloud it is recommended to enhance contrast and adjust exposition value accordingly, so the point cloud software is able better recognize control points in contrast pixels. It is noteworthy, that exporting images from photo editor, it is necessary to index all the images for later control in point cloud authoring tool. Adobe Lightroom was used for edition because it allows to create graphic presets and apply them to the whole frame package at once exporting it with the proper index. (Figure 2.22)

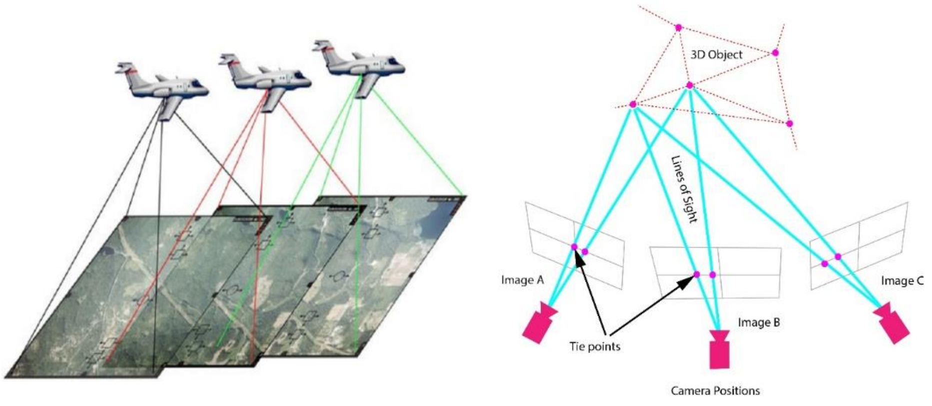


Figure 2.19 - Photogrammetry principal scheme

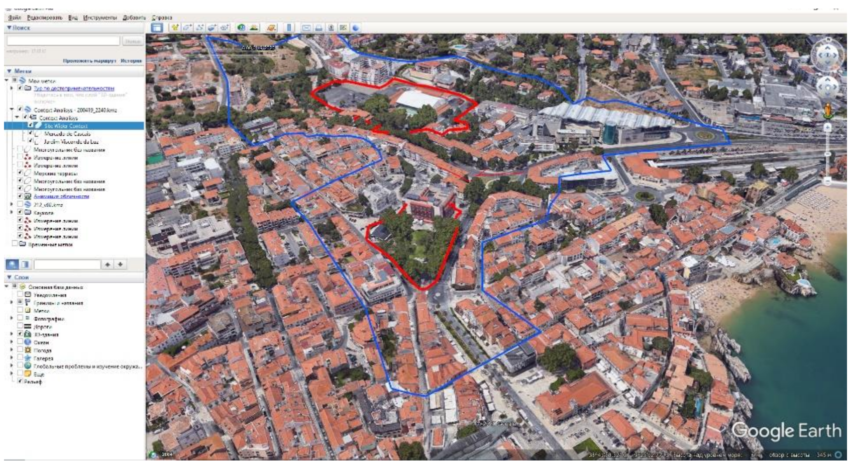


Figure 2.20 - Google Earth capturing

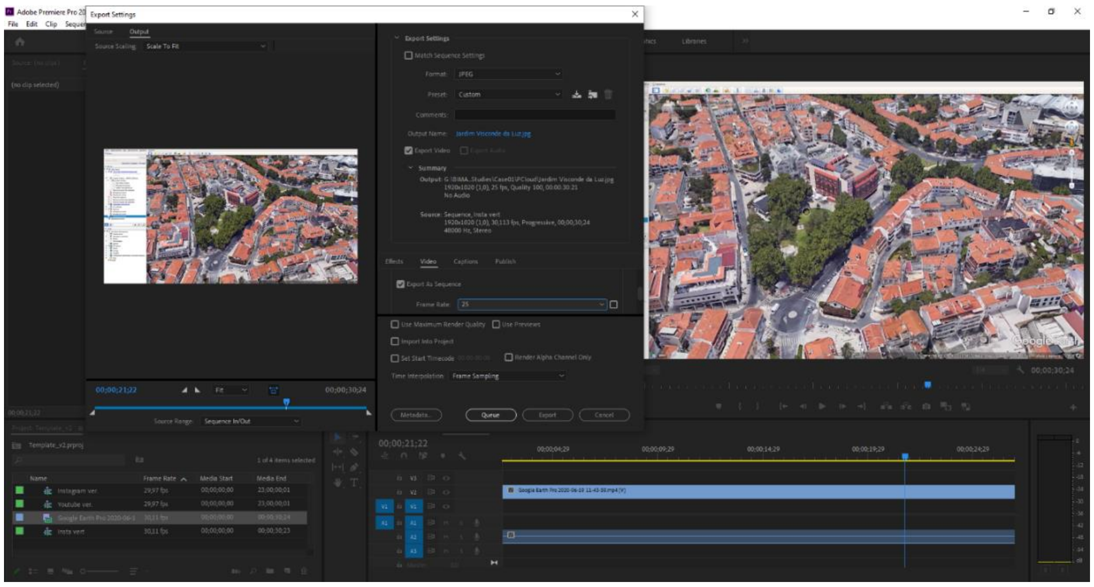


Figure 2.21 - Extracting frames from video record in Adobe Premiere Pro

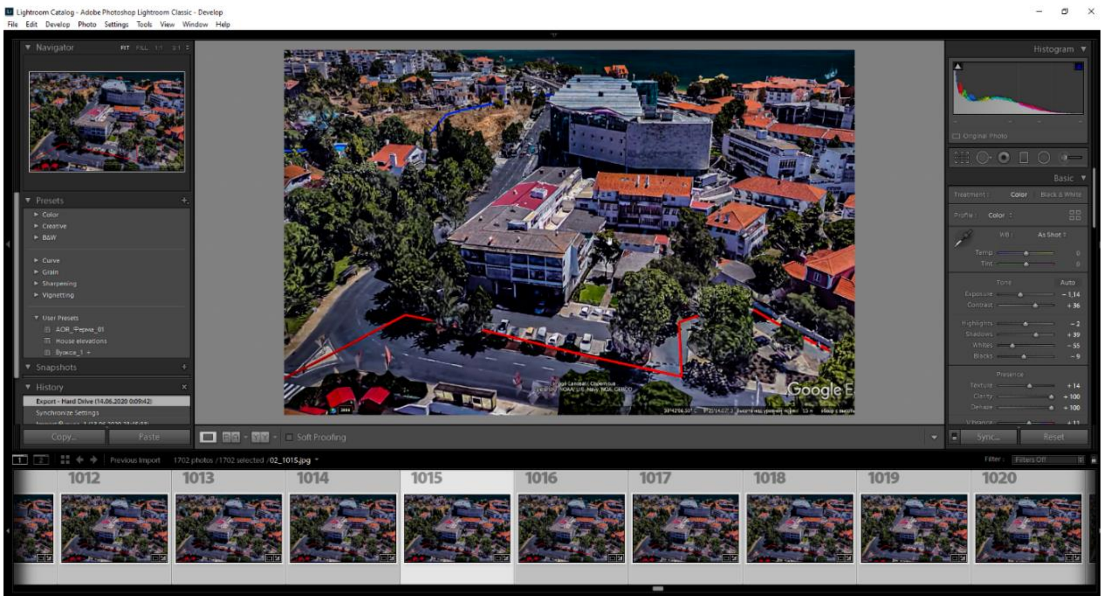


Figure 2.22 - Post-editing images in Adobe Lightroom

After edition process is completed, images are imported to RealityCapture and roughly aligned first. RealityCapture, as point cloud authoring tool, was chosen because it does not have image import limitation as it is in ReCap. Moreover, the tool provides deep control effortlessly. RealityCapture automatically aligns all of the imported images and then reconstructs a 3D model, textures and allows to inspect and adjust the point cloud quality. This takes a few minutes of manual work and a few hours at most of the background compilation by the RealityCapture software.

When draft alignment is performed, the low-resolution point cloud is assessed visually for defects. If such defects are detected, Ground Control Points (GPCs) are manually added by picking on a set of overlapping images. By recommendations from Pix4D guide, at least three images need to be defined for computation and estimation coordinates of the GCPs. These estimated GCPs are further reprojected in all the images where it might be visible in. (Figure 2.23)

After adding GCPs for correction of defected zones, final alignment is performed and a high-resolution point cloud is reconstructed within a user-defined region. Reconstructed point cloud has three display modes: vertices, solid, and sweet (textured solid) and can be exported either as a mesh or the point cloud. The reconstructed point cloud is further checked for topology defects and edited with such tools as Simplify Tool, Smoothing Tool, Close Holes, Clean Model, and Filter Selection.

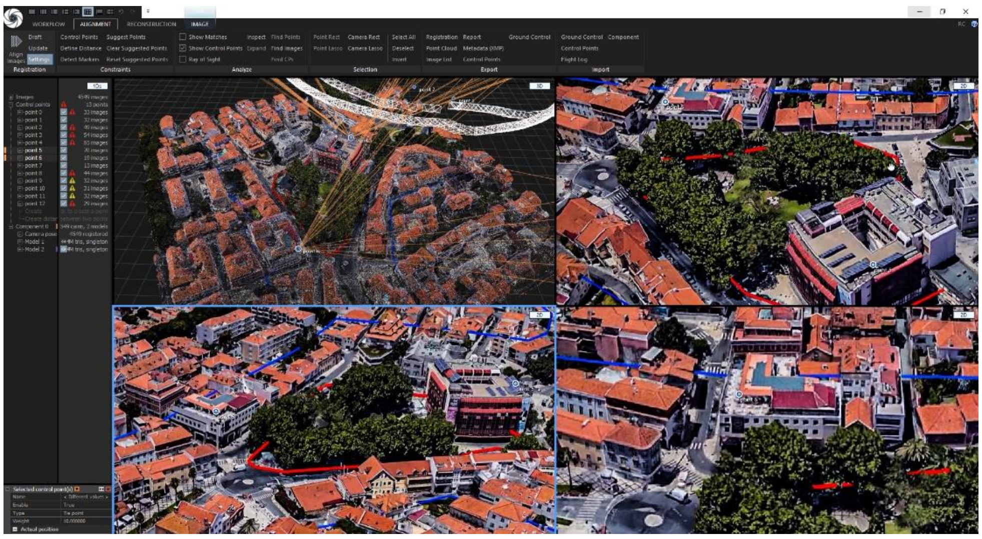


Figure 2.23 - Adding control points for point cloud alignment

Next step is unwrapping mesh for texturing. Unwrap tool has several settings such as maximum texture resolution, gutter, large triangle removal threshold, style, and texel (texture pixel) size. The main goal of adjusting these parameters is an achievement of the highest possible texture utilization which can be found in reconstructed model properties.

The last step is texturing of the point cloud and export it in format .xyz. All the steps processing point cloud in authoring tool takes awhile, including alignment, geo-referencing with GCPs, reconstruction, unwrapping, and texturing. Among these processes, alignment, reconstruction and texturing are the most time-consuming. A good practice to reduce the overall time is to divide images by sets. Division principle is usually the same as in Photogrammetry on-site. According to Photogrammetry Naming Convention (RealityCapture, 2020), Sets can be formed by flight altitude, flight type (Aerial Grid, Aerial Ring, Aerial Orthos - elevations), and image view angle. After division on sets, it is recommended in RealityCapture manual to run alignment for each set and create separate components on the first stage, then align components. Such workflow is faster than the alignment of all of the images at once.

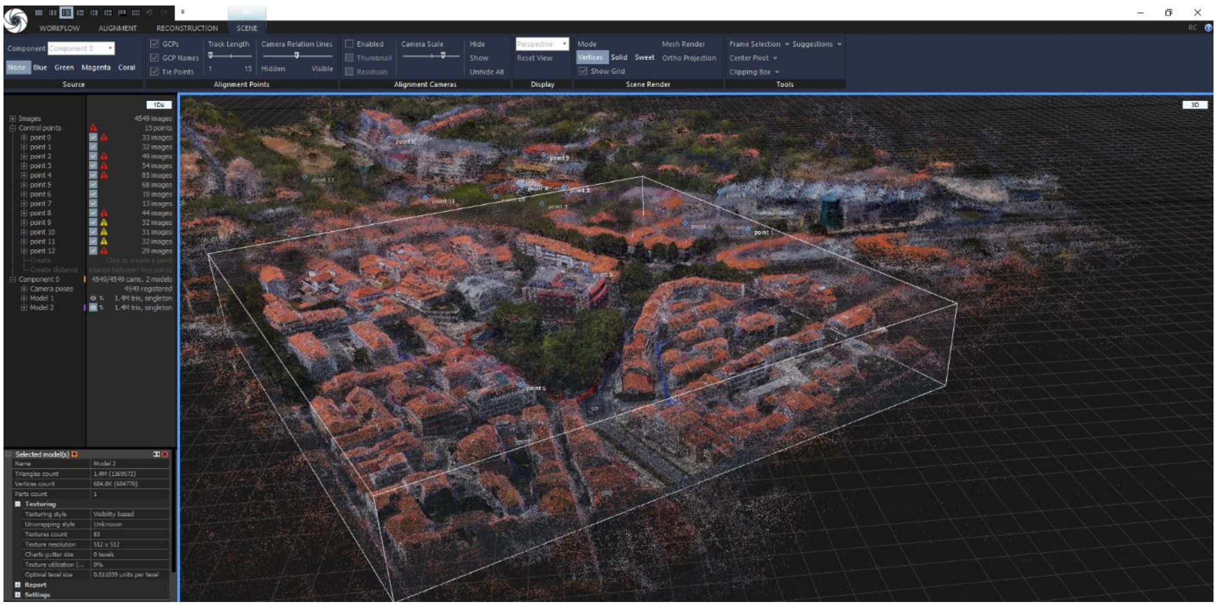


Figure 2.24 – Final alignment with GCPs and Reconstruction Region defined

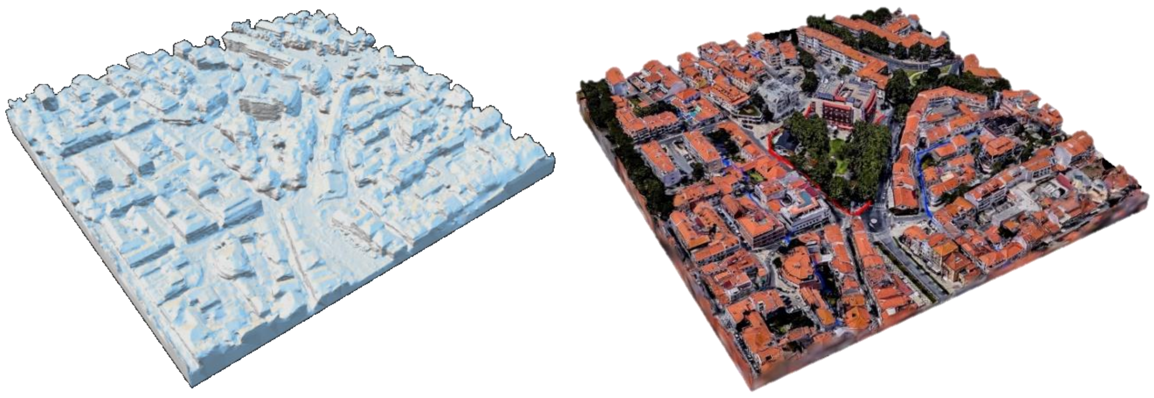


Figure 2.25 – Reconstructed point cloud in solid mode (left), after texturing (right)

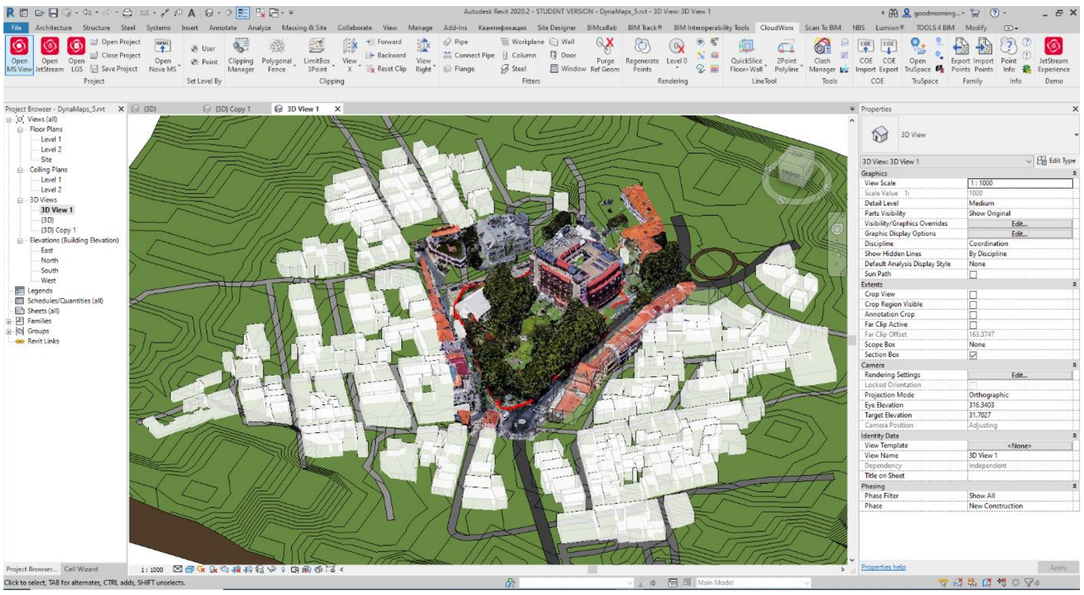


Figure 2.26 – Point Cloud linked, simplified, and cropped in the Site Model (Revit)

Consequently, the point cloud can be exported to BIM authoring tools such as Revit or ArchiCad or to 3d rendering tools such as Lumion or Twinmotion. If the objective is to assess design then BIM authoring tool is a choice. Export format, in this case, is .xyz. Point cloud then imported to ReCap and saved in .rcs or .rcp format readable in BIM authoring tool (Revit). The difference between formats is that .rcp is ReCap file of the project storing just project metadata with reference to point cloud. Meanwhile, the point cloud itself in unified .rcs format. Once the point cloud is linked and aligned to the BIM model, it can be adjusted with tools such as ScanToBIM and Leica CloudWorx for further integration needs. ScanToBIM allows to crop point cloud, extract some elements such as walls, pipes, and ducts. Another plugin Leica CloudWorx for Revit provides a virtual visit to the site within Revit with a complete view of the captured reality. Moreover, Leica plugin provides accurate fitting of steel, flanges, pipes, and 2D lines, or placement of built-in families such as walls, floors, structural framing, doors, windows, mechanical equipment, etc. As a result, the point cloud can be acquired without intervening the construction site and be integrated into the BIM model for the model quality assurance and basic families generation. Quite wide range of tools for working with point clouds already exist within BIM authoring tools and continues to develop.

If the objective is design visualization in the context, then the 3d rendering tool is the right option. Hence, .fbx or .obj formats are required. In the case study, Lumion considered for rendering as fast and simple real-time renderer. For import point cloud as solid textured geometry, Lumion reads format .fbx which can be exported straight from RealityCapture. (Figure 2.27) The design geometry exported in the same file format from Revit and stays synchronized since it is integrated now with Lumion. Thus, the workflow makes possible design authoring keep developing meanwhile updating automatically and visualizing it in Lumion.

As seen from above, the point cloud captured completely online can be utilized as a stand-alone context model for visualization tasks as well as integrated for the design authoring purposes. Meanwhile, the quality of such point cloud is limited by source Google Maps 3d geometry. Nevertheless, it is a relatively fast and effective method for preliminary site analysis and visualization.

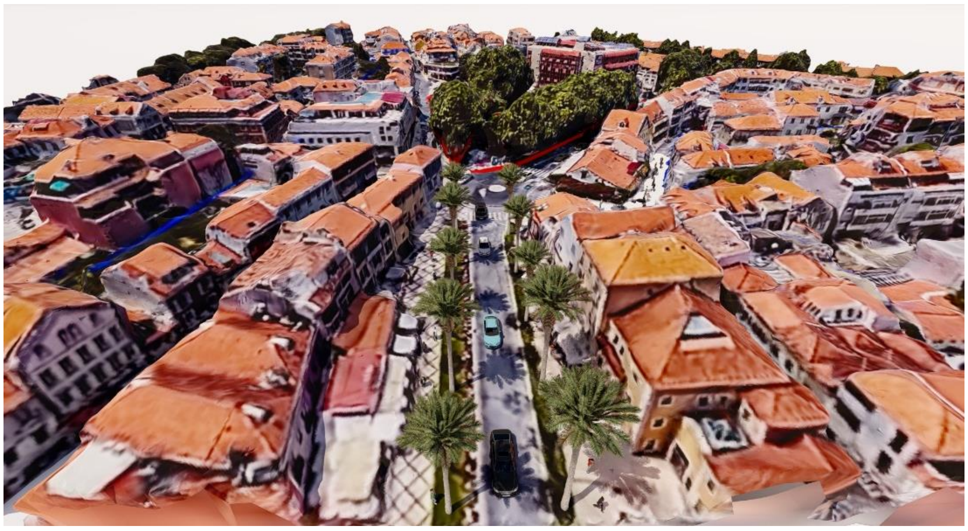


Figure 2.27 - Point Cloud imported in Lumion for render

2.3. CONCLUSION.

In conclusion, context capture and analysis integrated within a BIM environment is an essential feature for authoring design during preliminary studies. Appropriate workflow for Site Model generation was investigated using algorithm-aided design tools such as Elk and DynaMaps based on Dynamo as a visual programming platform. Besides different ways of context capturing were explored, particularly extracting and interpreting data from open sources such as OSM and USGS. Moreover, online capturing context geometry through Google Maps with photogrammetry technology is investigated as well.

As a result, Site Model was generated in two milestones. First, the topography was extracted and baked from OSM using DynaMaps tool. Then, the context geometry was generated with consideration of constrains and filters such as the distance to the construction site. Context geometry includes buildings and roads projected on toposurface. Based on Site Model, Site Analysis was performed. Additionally, Point Cloud was captured online using photogrammetry approach and integrated into BIM authoring software – Revit, for design assessment, as well as into 3D Visualization platform – Lumion, for presentation purposes. Generally, the workflow defined in the case study allows working with the context within a wide range of scenarios without visiting the site. Such methodology shifts design abilities forward. Though it still does not fully substitute the on-site capturing taking into account limitation of the data quality available online. Nevertheless, such workflow demonstrates a huge potential as a cheap and fast method for Site Model related design activities. We believe that in soon, databases will be fulfilled with the necessary datum.

This page is intentionally left blank

3. CASE 02. DOCUMENTATION AUTHORING

3.1. INTRODUCTION

Despite the development of presentation techniques and supports, project documentation still relies greatly on paper communication, although often this is already of digital format. Despite rapid digitalizing the AEC industry, we still inherit paper organization principles for documentation authoring. For this reason, much of the architect's time is still spent organizing a paper structure and placing drawings on a sheet. The traditional paper drawing is an additive process, in which complexity is achieved by the addition and overlap of independent signs. No associative relations can be managed between such signs as tags and dimensions. The internal consistency of drawing is not guaranteed by the medium but is entrusted to the designer. (Tedeschi, 2014) As a result of the investigation, a set of problems arose in the traditional drawing: low trust because of human factor; inefficiency in multiple iterative operations peculiar to documentation authoring; hard traceback and backlog managing; information loss. Nevertheless, current BIM tools allow us to build the information model and use sheet and view spaces as a smart medium with generated consistent relationships between elements for presentation. Architects, engineers, and designers can benefit from interaction with the BIM model through systematic managing and scripting its relationships to generate trustful documentation effectively and with less information loss.

ТЕХНОЛОГИЯ СОЗДАНИЯ ГЕОМЕТРИИ
ОТДЕЛКИ ПОМЕЩЕНИЙ В REVIT + DYNAMO
ВИДЕО 1. ФОРМИРОВАНИЕ ГЕОМЕТРИИ
ОТДЕЛКИ ПОЛА С УЧЕТОМ ГЛУБИНЫ
ДВЕРНЫХ ПРОЕМОВ
AUTODESK EXPERT ELITE
AUTODESK
R
Dynamo
IND ARCHITECTS

Figure 3.1 Room finishing script

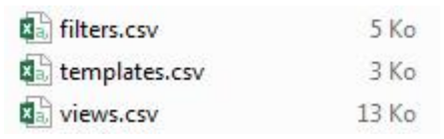
There is plenty of scripting implementation for solving documentation authoring issues asserting this statement. Automatic generation of frequentative elements is one of the essential tasks for detail drawings in BIM. For instance, the room finishing generation was scripted by Ind Architects, Russia (Figure 3.1). The set of scripts saves much time modelling all the elements manually and eliminates possible mistakes through prescribing behaviour during authoring design. So, by selecting the room and type of finishing script can generate all the finishes if the determined type.

Another example is sorting information by belonging to any asset using Design Script and Python methods unavailable by default. It can be floors, ceilings, furniture, or mechanical equipment grouped by room or zone for farther analysis or quantity take-offs. One of the methods is a test for the inclusion of the element location point in asset volume extracted as geometry from group entities.

Moreover, machine learning is already implemented in automatic design verification with tools such as UpCodes AI. The tool automatically scans the model for building code violations and gives a Summary of the code violations and requirements in simple language. Each instance of the violation is listed, noting the exact area or dimension causing the violation. It is possible to select and isolate each instance for further investigation. The report is also provided with snippets from the full section of original code text allowing to jump to the code directly. Significantly, the Codes database stays up to date to ensure specialists navigate the latest building codes without the need to reinstall or update tool (Figure 3.2). Such tools help to deliver more accurate drawings to the documentation examiner and reduce the number of revision cycles before project approval. Though such tools are still region-based and code updates highly dependent. Therefore, it fits for a general analysis of most common violations only, and it is not a replacement for professional code consultants yet. However, it is straightway forward artificial intelligence full integration within BIM.

Figure 3.2 – UpCodes AI tool for Revit BIM Code verification

One more important milestone in implementation scripting is model documentation management. This issue can involve some tedious tasks. For example, cleaning up the model after import all categories of an old Revit model is one of the tedious of these tasks. When someone import elements from another model, we quickly end up with thousands of view templates, filters, and other user-created views which can become unmanageable. To help address this problem, the script definition was written by Simon Moreau in 2015. (Figure 3.5) The script exports every view, template, and filter to three CSV files accordingly (Figure 3.3). To read these files in a meaningful way, PowerPivot is used in Excel to create a relational database (see Glossary), with two exported relational tables. (Figure 3.4) Once loaded in the PowerPivot tool, this data allows to quickly identify which template or filter are used and delete the unwanted ones. (Figures 3.6, 3.7) Besides, documentation, gathered by BIM manager and established in BIM Execution Plan, can be translated to the model from Excel, directly creating levels, building coordinates, and defining project information. It can be a proper setup for the project on the early stage of design.



filters.csv	5 Ko
templates.csv	3 Ko
views.csv	13 Ko

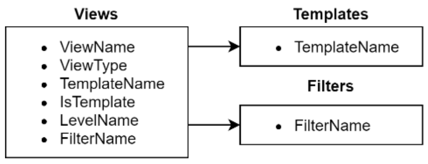


Figure 3.3-Exported CSV files from Revit

Figure 3.4-Relational Database in PowerPivot (Excel)

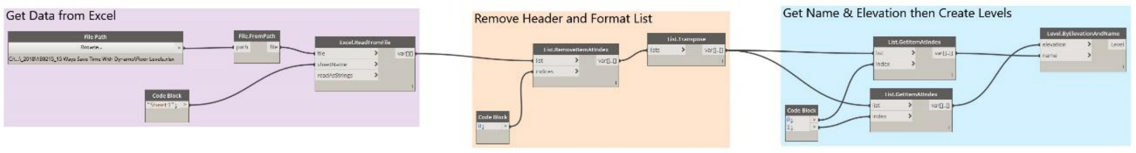
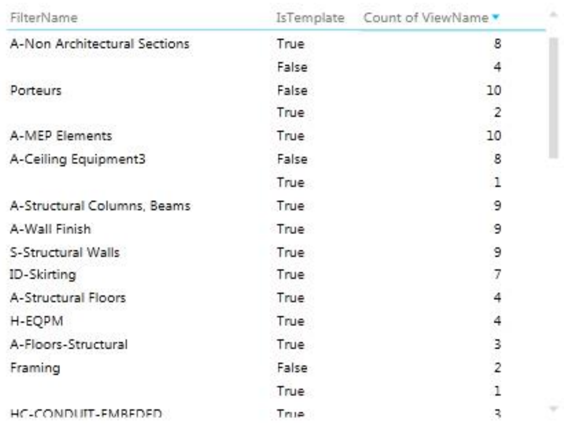
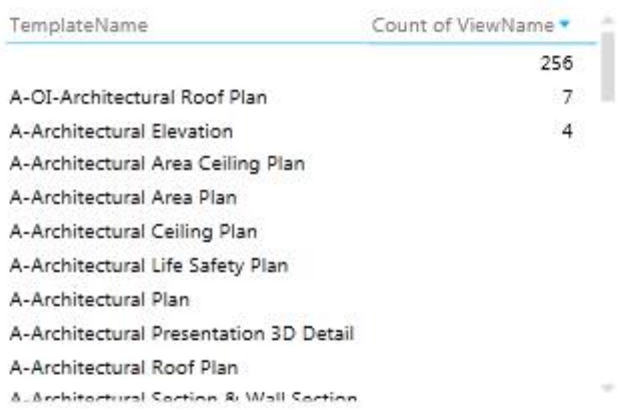


Figure 3.5 – Dynamo script translating data from Excel to Revit and creating Levels

Generating reports and quantity take-offs to share with the owner or other consultants is another good example. Scripting helps to control parameter interdependencies, calculate values, and exchange information reliably and interactively. For instance, the script can export to Excel all the instance and type parameter values for each instance of the selected family to generate FFE (furniture, fixtures, and equipment) spreadsheets. (Figure 3.8)



FilterName	IsTemplate	Count of ViewName
A-Non Architectural Sections	True	8
	False	4
Porteurs	False	10
	True	2
A-MEP Elements	True	10
A-Ceiling Equipment3	False	8
	True	1
A-Structural Columns, Beams	True	9
A-Wall Finish	True	9
S-Structural Walls	True	9
ID-Skirting	True	7
A-Structural Floors	True	4
H-EQPM	True	4
A-Floors-Structural	True	3
Framing	False	2
	True	1
HC-CONDUIT-EMBEDED	True	3



TemplateName	Count of ViewName
	256
A-OI-Architectural Roof Plan	7
A-Architectural Elevation	4
A-Architectural Area Ceiling Plan	
A-Architectural Area Plan	
A-Architectural Ceiling Plan	
A-Architectural Life Safety Plan	
A-Architectural Plan	
A-Architectural Presentation 3D Detail	
A-Architectural Roof Plan	
A-Architectural Section & Wall Section	

Figure 3.6–Filter Table (PowerPivot) **Figure 3.7-Template Table (PowerPivot)**

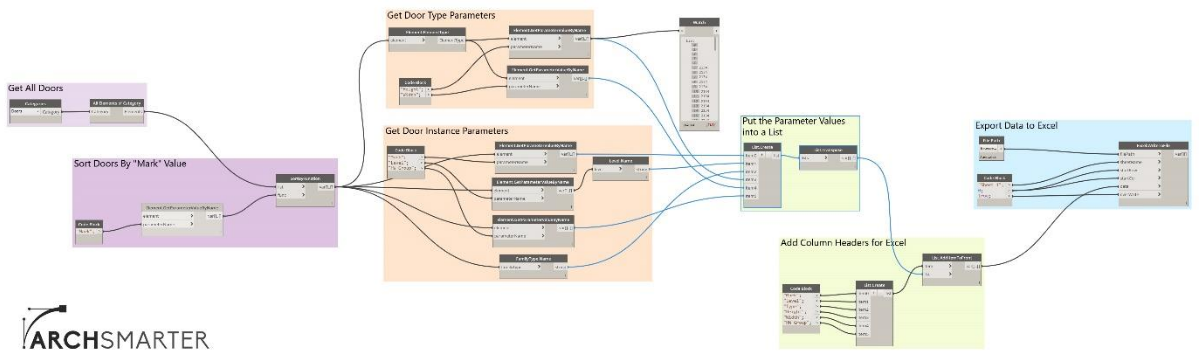

Get Door Type Parameters
Get All Doors
Sort Doors By "Mark" Value
Get Door Instance Parameters
Put the Parameter Values into a List
Export Data to Excel
Add Column Headers for Excel
ARCHSMARTER

Figure 3.8 – Dynamo script exporting instance and type parameter values to Excel

At present, drawings are not considered anymore as a product of a process involving manual human gestures for input signs. Instead, drawing is described as a section view of the model with linked tags and parametric elements cropped within the view range with control parameters interconnecting all the presences of these elements within the scope of such views. Consequently, such an approach allows us to script elements presence and their attributes as well as view attributes controlling the whole model and its behaviour centrally by own prescribed definitions. The well-known example is the auto-dimension issue. It takes much time to draw all the dimensions manually and requires a sharp review along the design authoring lifecycle. To solve this problem, the script was elaborated by Michael Kilkelly within the scope of his script library. The script automatically dimensions vertical and horizontal grid lines in a view and exempt of need to click every gridline manually. This script does it all in a few seconds (Figure 3.9).

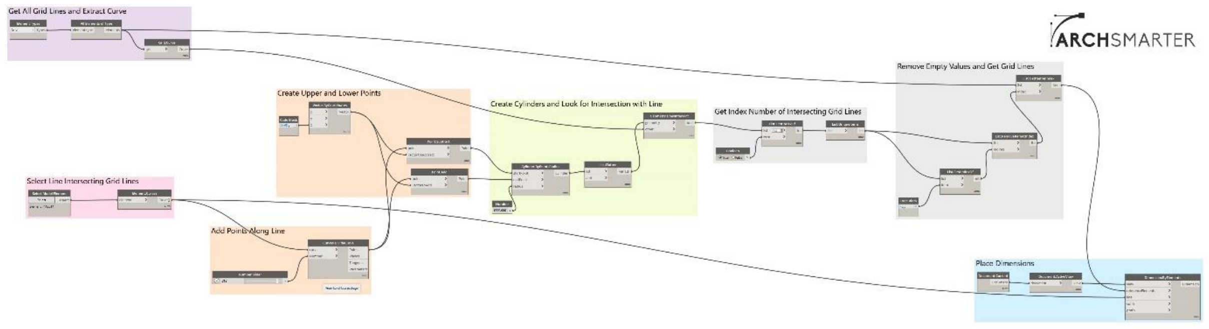

Get All Grid Lines and Extract Curve
ARCHSMARTER
Remove Empty Values and Get Grid Lines
Create Upper and Lower Points
Create Cylinders and Look for Intersection with Line
Get Index Number of Intersecting Grid Lines
Select Line Intersecting Grid Lines
Add Points Along Line
Place Dimensions

Figure 3.9 – Dynamo script for auto-dimension

As seen above, AIM tools are widespread for solving complex issues and accelerating design documentation authoring in general. Among the scope of issues the most essential are the generation of frequentative elements, sorting and grouping data, design verification, model data management and exchange, generation of reports and quantity take-offs, and graphical views control.

Further, within the case scenario proposed by the partner company, automatic views generation for compartments and their organized layout on the sheet are considered.

3.2. CASE STUDY. VIEWS GENERATION AND AUTOMATIC LAYOUT

Compartments or room objects are common assets in the Building Information Model and often used for sorting information, specifications, and views accordingly. Usually, the scope of views with information attached is predefined, especially in typical projects with iterative design inclusions. Hence generation of asset referred views, specifications, and attributes, as well as their layout on sheets, is a vital part of documentation authoring. Meanwhile, this part can be automated through scripting and open

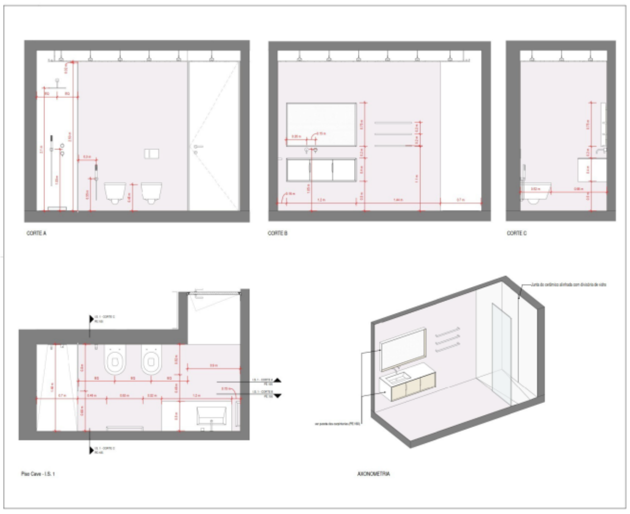


Figure 3.10 - Illustration of a composite asset sheet showing floor plan, internal elevations and axonometry

API. Asset scope often consists of the floor plan, ceiling plan, elevations, and axonometric view. Afterward, the views are placed into sheet space according to asset belonging. (Figure 3.10)

Then the designer makes a manual layout for composition on sheet. Although a sheet layout may differ greatly because of the project being communicated, common features can be defined among most of the projects compartments/ room. The intention is to define a workflow and subsequent routines that lay on a sheet a set of predefined views by room.

Following the above request, workflow including two scripts generating a set of views by room and laying them out on sheet is created.

Primarily, the workflow was divided into two parts. The first part was dedicated to the generation of views and sheets out of room selection and default layout views on sheets. The second part presents a custom layout setup for created views on the chosen sheet.

To begin with, parametric and generative schemes were elaborated. The parametric scheme defines input parameters and the scope of output results. Generative scheme structures the principal generation algorithm and derives the whole definition into several single tasks with its inputs, bodies, and output objects and attributes. The parametric scheme consists of next parameters as input:

- Room Selector
- View attributes: Elevation Type, Crop Elevation Offset, Crop Plan Offset, Marker Offset from Wall
- Sheet attributes: Titleblock Type, Package
- Constraints: Min Wall Length for Elevation
- Booleans:CreateElevations, CreateFloorPlans, CreateCeilingPlans, Create AxonometricViews, Layout Type

For output, definition shell produce views laying out on sheets by room asset. Generative schema represented in 3 milestones and reflects the principle behind workflow. The workflow in its turn consists of two scripts. First, “CreateViewsByRoom”, generates views with defined attributes by the user and default placement on sheets. The second one, “LayoutOnSheet”, allows to customize the layout by type with a defined sequence of views and path for layout.

1. Reconstruction room boundary corresponding to views generation
2. Views creation
3. Sheets creation and layout

3.2.1. 1st milestone: reconstruction room boundary for views generation

Initially, all input parameters were defined, room and linked level filters were applied to extract necessary room attributes. Then the further part of the script is/was dedicated to the reconstruction of the room outline with filtering sides for elevations. Room boundary is/was enriched with a set of lines corresponding to the number of walls surrounding the room. The script intended to extract these lines as the basis for elevations. However, these lines are not applicable as the walls can be divided within a single room side. Nevertheless, they are essential for the whole side elevation generation. To handle this issue, a boundary reconstruction method was elaborated. The method assumes defining room corners and reconstruction boundaries by connecting these corners by new lines which would be the basis for the views. First of all, angles are found between consecutive lines pairs to determine corners. Consequently, corner lines are defined through filtering by an angle larger than 10 degrees meant to be room corners. Afterwards, corner points are specified by projecting the first line in the pair to the second. (Figure 3.11)

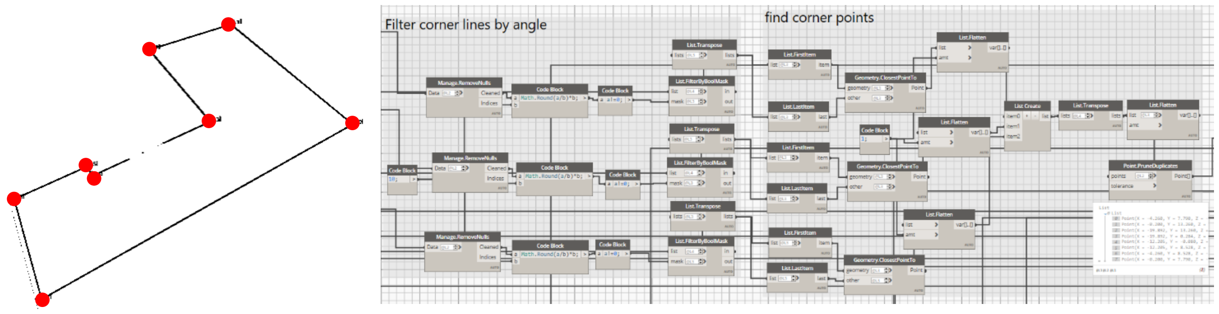


Figure 3.11–corner lines and points detected by script. Definition on the right (Dynamo)

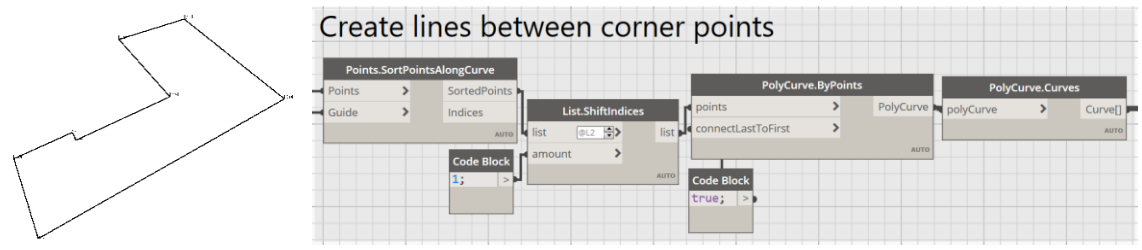


Figure 3.12–New boundary by connecting lines between corner points

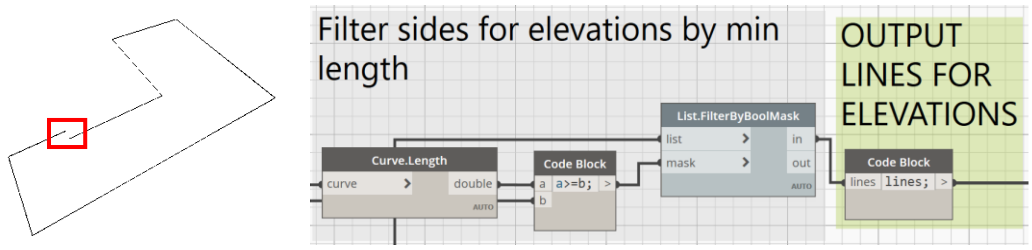


Figure 3.13 – Filter boundary sides for elevation generation. Filtered side marked by red

The next step was to create a new boundary consisted of lines defining crop box basis for future views. (Figure 3.12) After the lines were created, they got filtered by constraining min length for elevation generating. (Figure 3.13) The outputs of the first part are basis lines, determining elevations, which afterward will be input values for elevations.

3.2.2. 2nd Milestone: Views Creation

Views creation is divided into three different workflows meant separately for plan views (floor plan, ceiling plan), elevations, and axonometric view. Unlike plans and axonometric views which can be created fully by built-in Dynamo nodes, elevations accomplished by Python custom script with opening Revit API transaction. Despite there are dedicated nodes in Rhythm (CreateElevationByMarkerIndex) and LinkedIn Learning (ByScopeBox) Packages, both of them are not enough defined (with Scop Region in the first case, and Facade distance in the second) for automatic elevation creation by room.

Consequently, the farther part of the script is dedicated to extracting and interpreting information from room entities to create views in three different ways according to view types. For elevations, Python script is written. To run the script it requires Marker Points, Facade Middle Points, Crop Region Lines, and View Family Type elements for inputs. To begin with, elevation lines created on the last step and Facade Middle Points, which are middle points of boundary lines, are moved (translated) inside the room. (Figure 3.14) Translation controlled by input parameter "Marker Offset from the wall". Marker Points are moved Facade Middle Points.

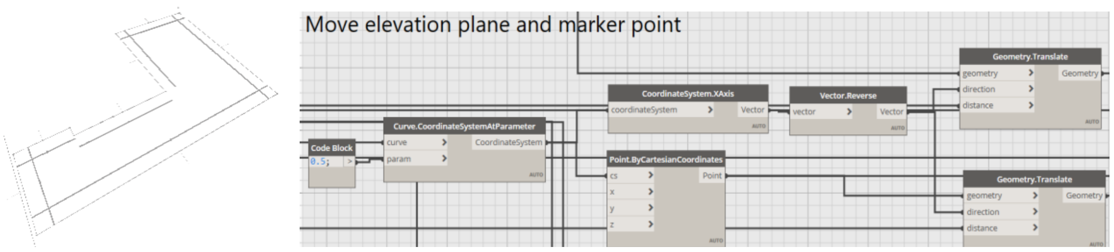

Move elevation plane and marker point
Geometry.Translate
CoordinateSystem.XAxis
Vector.Reverse
Curve.CoordinateSystemAtParameter
Code Block
0.5;
Point.ByCartesianCoordinates
Geometry.Translate

Figure 3.14 – Elevation base lines moved of the boundary

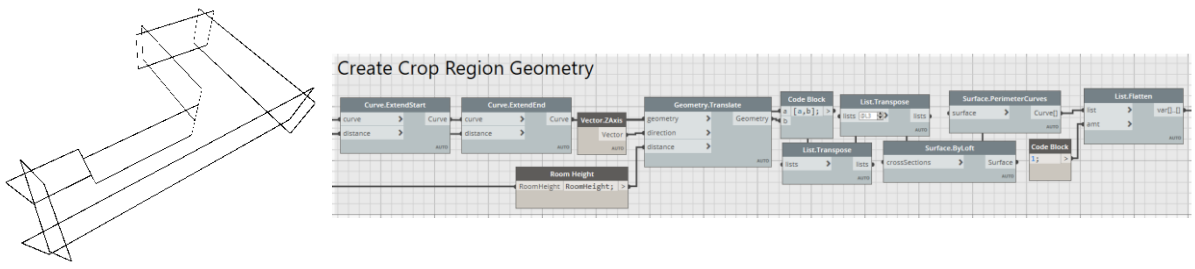
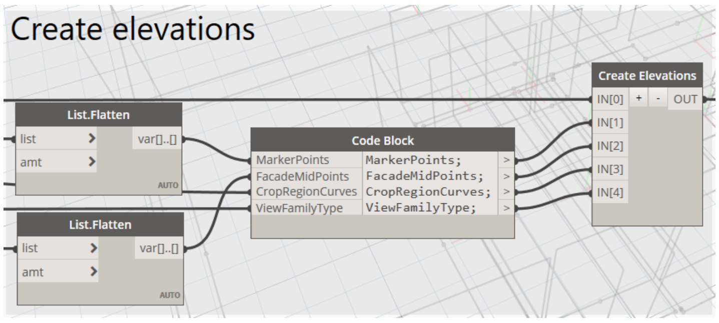


Figure 3.15 – Crop Region Lines for elevations

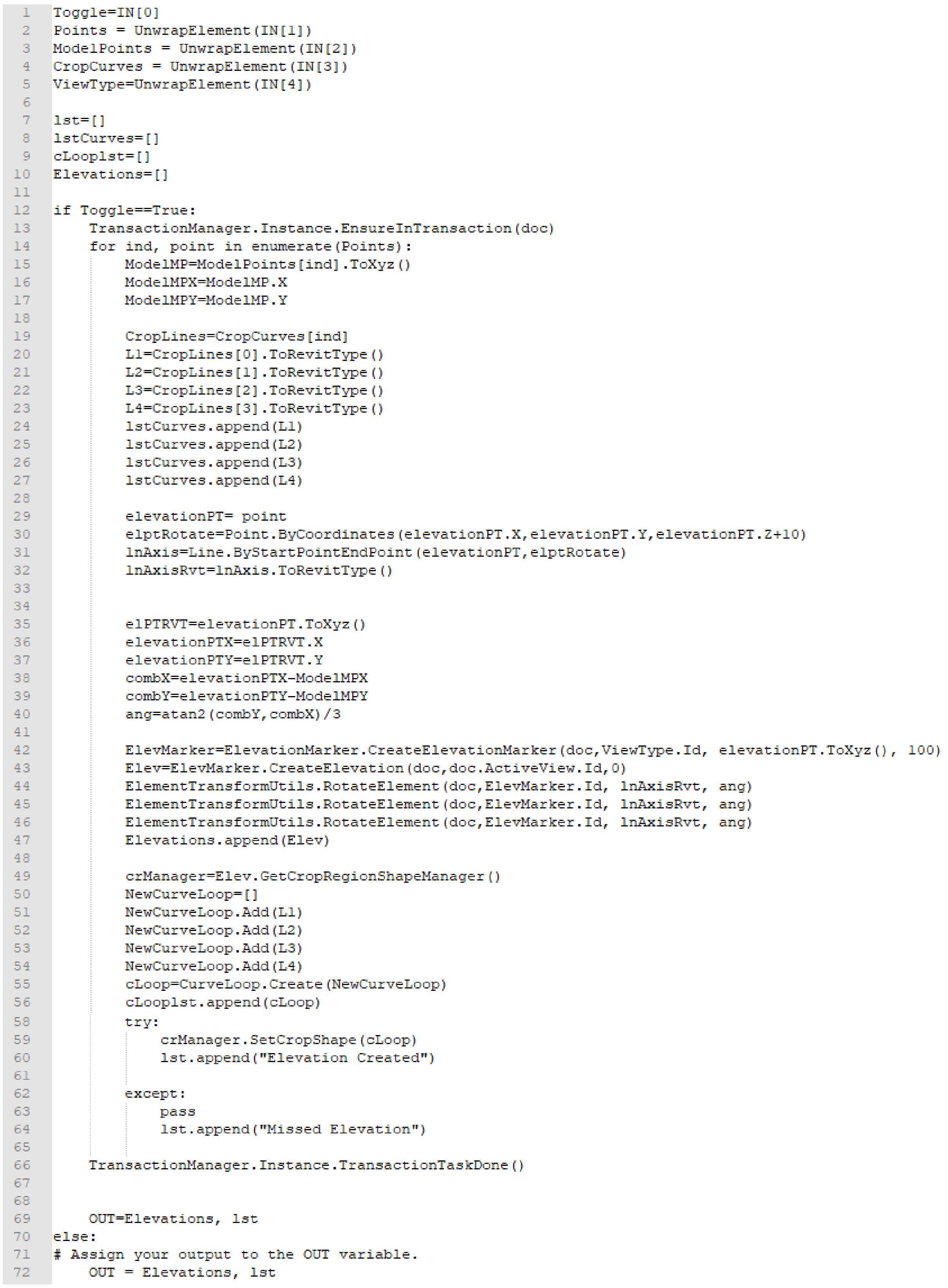
Secondly, the Crop Region Lines were defined by translating elevation lines up. Crop Region Offset is controlled by the input parameter "Crop offset aside elevations". (Figure 3.15)

The last step before run Python script was to define View Family Type which is selected by the user as input parameter. Then Python script was written in adoption from script shared by Jeremy Tammik (Tammik, 2016). Inputs are shown in Figure 3.16. Script unwraps elevations based on model lines of some sort. This works well especially for unwrapping large projects like stadiums with multifaceted facades as well as unwrapping interior elevations by walls or manual model lines. It can be used as well in the way where walls may be linked in from another model in which case default method for elevations will not snap to them. So the script is quite versatile. A further adapted script body is presented in Figure 3.17.



```
MarkerPoints;
FacadeMidPoints;
CropRegionCurves;
ViewFamilyType;
```

Figure 3.16 – Input values for Python script



```python
Toggle=IN[0]
Points = UnwrapElement(IN[1])
ModelPoints = UnwrapElement(IN[2])
CropCurves = UnwrapElement(IN[3])
ViewType=UnwrapElement(IN[4])

lst=[]
lstCurves=[]
cLooplst=[]
Elevations=[]

if Toggle==True:
    TransactionManager.Instance.EnsureInTransaction(doc)
    for ind, point in enumerate(Points):
        ModelMP=ModelPoints[ind].ToXyz()
        ModelMPX=ModelMP.X
        ModelMPY=ModelMP.Y

        CropLines=CropCurves[ind]
        L1=CropLines[0].ToRevitType()
        L2=CropLines[1].ToRevitType()
        L3=CropLines[2].ToRevitType()
        L4=CropLines[3].ToRevitType()
        lstCurves.append(L1)
        lstCurves.append(L2)
        lstCurves.append(L3)
        lstCurves.append(L4)

        elevationPT= point
        elptRotate=Point.ByCoordinates(elevationPT.X,elevationPT.Y,elevationPT.Z+10)
        lnAxis=Line.ByStartPointEndPoint(elevationPT,elptRotate)
        lnAxisRvt=lnAxis.ToRevitType()


        elPTRVT=elevationPT.ToXyz()
        elevationPTX=elPTRVT.X
        elevationPTY=elPTRVT.Y
        combX=elevationPTX-ModelMPX
        combY=elevationPTY-ModelMPY
        ang=atan2(combY,combX)/3

        ElevMarker=ElevationMarker.CreateElevationMarker(doc,ViewType.Id, elevationPT.ToXyz(), 100)
        Elev=ElevMarker.CreateElevation(doc,doc.ActiveView.Id,0)
        ElementTransformUtils.RotateElement(doc,ElevMarker.Id, lnAxisRvt, ang)
        ElementTransformUtils.RotateElement(doc,ElevMarker.Id, lnAxisRvt, ang)
        ElementTransformUtils.RotateElement(doc,ElevMarker.Id, lnAxisRvt, ang)
        Elevations.append(Elev)

        crManager=Elev.GetCropRegionShapeManager()
        NewCurveLoop=[]
        NewCurveLoop.Add(L1)
        NewCurveLoop.Add(L2)
        NewCurveLoop.Add(L3)
        NewCurveLoop.Add(L4)
        cLoop=CurveLoop.Create(NewCurveLoop)
        cLooplst.append(cLoop)
        try:
            crManager.SetCropShape(cLoop)
            lst.append("Elevation Created")

        except:
            pass
            lst.append("Missed Elevation")

    TransactionManager.Instance.TransactionTaskDone()


    OUT=Elevations, lst
else:
# Assign your output to the OUT variable.
    OUT = Elevations, lst
```

Figure 3.17 – Python script for room elevations creation based on boundary

For Plan Views including Floor Plan and Ceiling Plan, the built-in nodes were applied based on input Level parameters, FloorPlanView.ByLevel and CeilingPlanView.ByLevel accordingly.

For the generation of axonometric views, a built-in node was used with input eye point and bounding box, AxonometricView.ByEyePointTargetAndBoundingBox. The node requires an input eye point, target point, Bounding Box, name, and isolation Boolean. (Figure 3.18) Bounding Box was constructed by extruding room boundary by 1.2m above the floor. Next, the minimum point of the bounding box was accepted as the target point and maximum point scaled in Z direction by the same distance as the line between its projection and target point to form 45 degrees view angle. Eventually, axonometric views were created and isolated. (Figure 3.19)

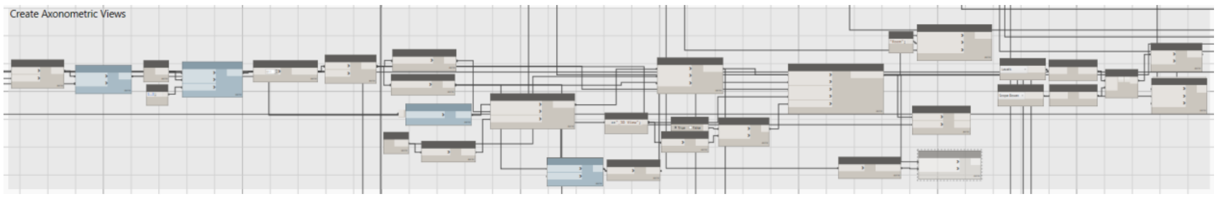


Figure 3.18 – Create Axonometric Views Definition

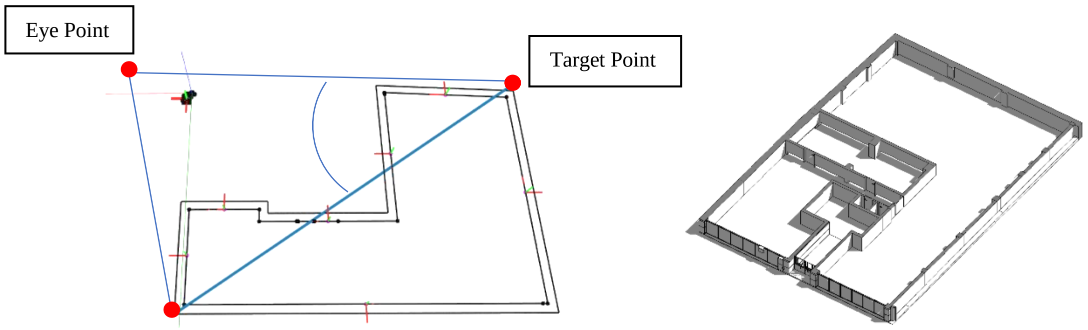
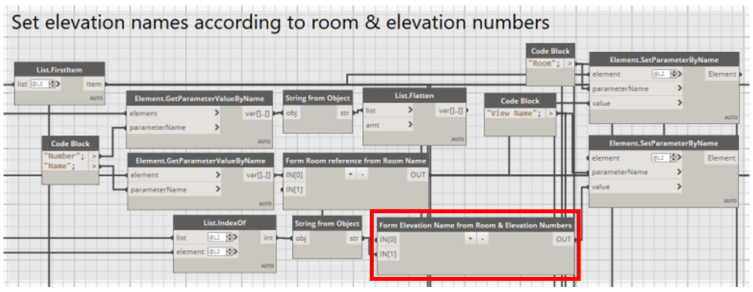


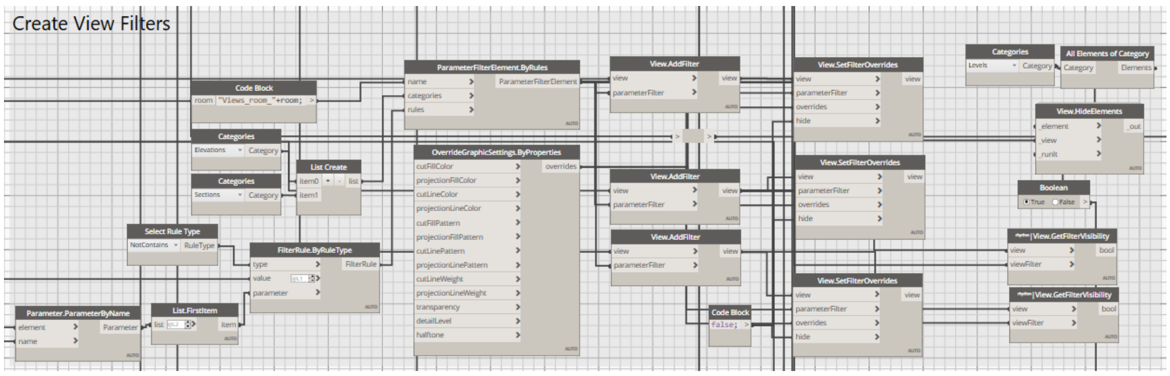
Figure 3.19 – Generative scheme and result axonometric view

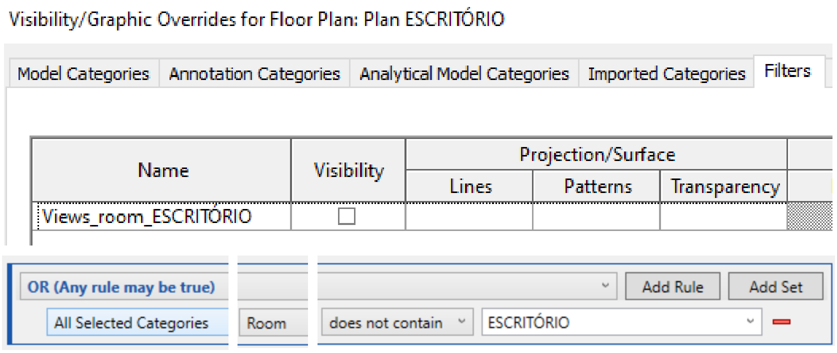
Further, no less important part of the script development is setting view names according to room & view indexes. (Figure 3.20) Special instance parameter “Room” for View Category was created to distinguish each view and group it for sheet layout and graphic filters. Python scripts were used again for the custom naming algorithm.


Set elevation names according to room & elevation numbers
List.FirstItem
Code Block
Element.GetParameterValueByName
String from Object
List.Flatten
Code Block
Element.SetParameterByName
Element.SetParameterByName
Form Room reference from Room Name
List.IndexOf
String from Object
Form Elevation Name from Room & Elevation Numbers

```
for i, rm in enumerate(RoomNum):
    for k, j in enumerate(ElevNum[i]):
        n=str(k+1)
        ViewName="Room Elevation "+rm+"-"+n
        ViewNameList.append(ViewName)
```

Figure 3.20 – Set elevation names and numbers according to room attributes. Python node in red. Enumerate loop part of Python script (on bottom)



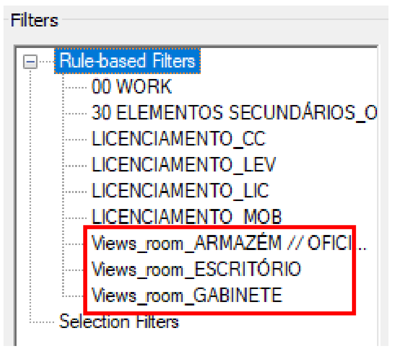

Visibility/Graphic Overrides for Floor Plan: Plan ESCRITÓRIO

Model Categories | Annotation Categories | Analytical Model Categories | Imported Categories | Filters

Name	Visibility	Projection/Surface		
		Lines	Patterns	Transparency
Views_room_ESCRITÓRIO	☐			

OR (Any rule may be true) | Add Rule | Add Set

All Selected Categories | Room | does not contain | ESCRITÓRIO


Filters

Rule-based Filters
- 00 WORK
- 30 ELEMENTOS SECUNDÁRIOS_O
- LICENCIAMENTO_CC
- LICENCIAMENTO_LEV
- LICENCIAMENTO_LIC
- LICENCIAMENTO_MOB
- Views_room_ARMAZÉM // OFICI...
- Views_room_ESCRITÓRIO
- Views_room_GABINETE

Selection Filters

Figure 3.21 – Filters creation and setting view overrides, Revit filter settings preview (on bottom)

Finally, filters are created and added to new views to hide the rest of the views and their tags which are out of the room. (Figure 3.21). Filter based on rule by containing Room parameter value for categories of sections and elevations. So, if the elevation view is not contained in the relevant room element, it gets hidden. Thus, every room element has its filter which added to relevant views.

3.2.3. 3rd Milestone. Sheet creation and layout

In the last part of the definition, sheets were created and prepared views were placed according to the uniform grid layout. In the beginning, all views created earlier were filtered and grouped by room name, empty sheets were created based on a title block type from existing types in the project. Then Sheet.Create node (Rhythm Package) run under recursion to create individual sheets per room. Figure 3.22)

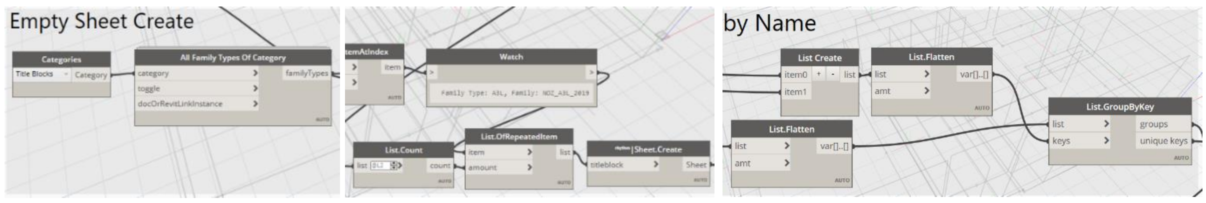
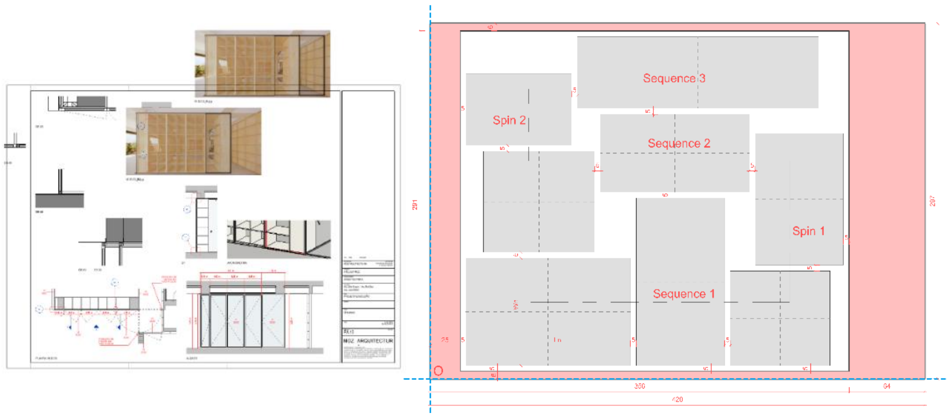


Figure 3.22 - Sheets creation and views grouping by room

Next, to create a layout, three different strategies were tested: sequential, nesting, and uniform grid layout. The sequential layout assumes a sequence of sorted views with a defined start position and path generated within the view placement area (the white area inside red boundaries in Figure 3.23). Once the item in sequence does not fit inside the placement area it goes to the next spin. Spins can be ruled by different algorithms. In the following example (Figure 3.23) each spin rotates path by 180 degrees. Another way could be to shift spins to a position above the first item of the previous spin. After every spin, algorithm tests for intersections with previous spin members. If such intersections exist, viewports are moved upward until there is no overlap. The weakness of such an approach that it is highly dependent on view scale and sheet size. If both parameters don't suite together, viewports will be overlapping otherwise leave gaps and hard to manually adjust afterward. Hypothetically, optimization or evolutionary solver, such as Refinery, could help to match views locations with their scale and sheet size. However, optimization would take a time in the best case comparable with the manual layout depending on hardware running. The problem would be how to maintain all the elements outside of the others. It is possible but very demanding solution.



```
WidthList = IN[0]
LengthList = IN[1]
Xs,Ys,dif = [],[],[]
Cp=[]
Cpl=[]
x = LengthList[0]/2+0.03
y = WidthList[0]/2+0.011
k = 1
spin=1
rt=0
Dx=0
Clx=0
BrMax, BrMin=0,0
Xs.append(x)
Ys.append(y)

for n in range(1, len(WidthList)):
  BrMax=0.356-(LengthList[n]+LengthList[n-1]/2+0.01)*k
  BrMin=0.025+LengthList[n]+LengthList[n-1]/2+0.01
  if x > BrMax:
    x=0.356-LengthList[n]/2-0.005
    y=Ys[n-1]+WidthList[n-1]/2+WidthList[n]/2+0.005
    k*=-1
    spin=1
    rt+=1
  elif x < BrMin and spin==0:
    x=0.025+LengthList[n]/2+0.005
    y=Ys[n-1]+WidthList[n-1]/2+WidthList[n]/2+0.005
    k*=-1
    spin=1
    rt+=1
  elif rt>0:
    x=Xs[n-1]+(LengthList[n-1]/2+LengthList[n]/2+0.005)*k
    for i in range(len(Xs)):
      Dx=abs(Xs[i]-x)
      dif.append(Dx)
    def closest(lst, K):
      return lst[min(range(len(lst)), key = lambda i: abs(lst[i]-K))]
    Clx=closest(Xs,x)
    y=WidthList[n]/2+Ys[Xs.index(Clx)]+WidthList[Xs.index(Clx)]/2+0.005
    spin=0
    Cp.append(Clx)
    Cpl.append(Xs.index(Clx))
  else:
    x=Xs[n-1]+(LengthList[n-1]/2+LengthList[n]/2+0.005)*k
    y=WidthList[n]/2+0.011
  Xs.append(x)
  Ys.append(y)

OUT= [Xs,Ys]
```

Figure 3.23 – Sequential layout: result (left), scheme (right)

Another possible layout is nesting. Nesting layout is the problem well known as a 2-dimensional packing problem. There is plenty of methods to solve this problem. (Figure 3.24) One of them is Strip Packing algorithms family. In the two-dimensional Strip Packing problem, we are given a strip of a finite width W but infinite height, and a set of rectangular items each of width at most W. The objective is to pack all the items into the strip to minimize the height used. The items may neither overlap nor be rotated. A common approach is level-oriented, the items are packed from left to right, in rows forming levels. Within the same level, all items are packed so that their bottoms align. The first level is the bottom of the strip and subsequent levels are defined by the height of the tallest item on the previous level. Known Strip Packing algorithms are First-Fit Decreasing Height (FFDH) algorithm, Next-Fit Decreasing Height (NFDH) algorithm, Best-Fit Decreasing Height (BFDH) algorithm, Bottom-Left (BL) Algorithm, Baker's Up-Down (UD) algorithm (B.S. Baker, 1981), Reverse-fit (RF) algorithm (Steinberg, 1997), Steinberg's algorithm (Steinberg, 1997), Split-Fit algorithm (SF) (Tarjan, 1980), and Sleator's algorithm (Sleator, 1980).

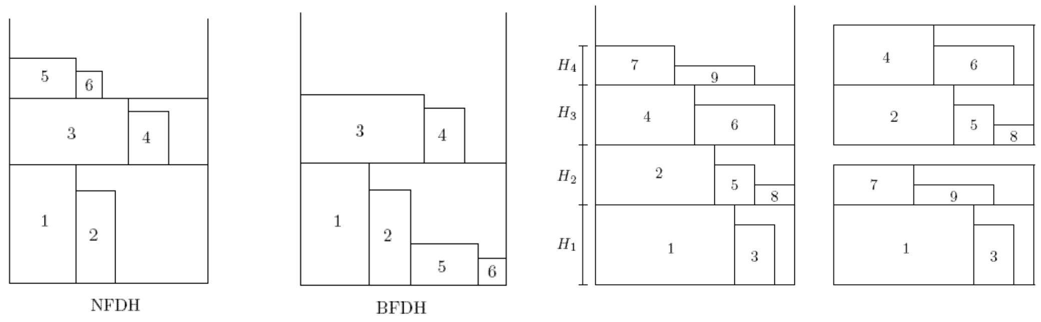


Figure 3.24 – Packing Algorithms. NFDH, BFDH are Strip Packed, HFF (right) is 2-phase Bin Packed add source

Another approach to 2-dimensional problem is Bim Packing algorithms. In the two-dimensional Bin Packing problem, we are given an unlimited number of finite identical rectangular bins, each having width W and height H, and a set of n rectangular items with width w_j <= W and height h_j, for 1 <= j <= n. The problem is to pack, without overlap, all the items into the minimum number of bins. The items cannot be rotated. Most of the off-line algorithm in the literature are of greedy type, and can be classified into two families:

- one phase algorithms directly pack the items into the finite bins; (eg. Finite Next-Fit (FNF), Next Bottom-left (NBL), Alternate Directions (AD) (Vigo, 1999))

- two-phase algorithms start by packing the items into a single strip, i.e., a bin having width W and infinite height. In the second phase, the strip solution is used to construct a packing into finite bins. (eg. Hybrid First-Fit (HFF) – Figure 3.24 on the right (F.K.R. Chung, 1982), Hybrid Best-Fit (HBF) (Wong, 1987), Floor-Ceiling (FC) algorithm (Vigo, 1998))

There are existing solutions for Packing Problem within Dynamo Packages library, such as Miscellany PackingService.PackContainer Node and Refinery Toolkits Packing.PackRectangles Node. However, the weakness, unlike the sequential layout approach, is that viewports which are out of placement space, get filtered. Therefore, filtered viewports need to be added semi-manually. Moreover, these methods don't have orientation constrains. Thus, in its current edition, it isn't possible to constrain the X and Y dimensions to their original axes. Consequently, Python script with loop and custom constrains following one of the mentioned above algorithms can be only a solution for nesting views currently. There are some available Python modules such as a rectangle-packer written by Daniel Andersson. (Andersson, 2019) Rectangle-packer, given for input a set of rectangles with fixed orientations, finds an enclosing rectangle of the minimum area that contains them all with no overlap. (Figure 3.25) However, to load external modules in Python node in Dynamo would cause interoperability issues afterward. Hence, running a script on another PC would become impossible unless the user will install manually all the modules involved in the script that is not versatile. Though it is possible to prepare a distribution package and setup.exe file.

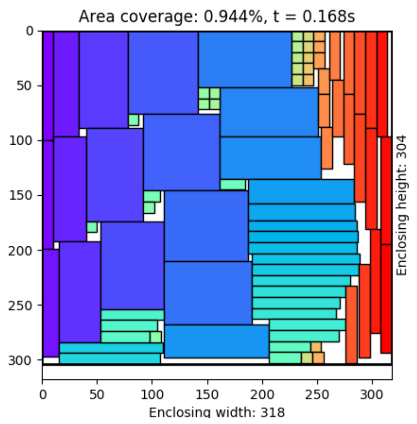


Figure 3.25 – Rectangle-Packer Python-module by Daniel Andersson

Eventually, Uniform Grid Layout became an approach of choice. The algorithm refers to a regular grid as a basis for viewport placement. Cell size defined by the largest viewport dimensions in both axes and the grid is independent of sheet size. Thus overlap is excluded by default and, despite it does not fit the sheet size, the structure of layout is clear and adjustable afterward. Two options are considered according to a vertical and horizontal layout. As additional input controls, options of horizontal and vertical layouts, Start Placement Point, and Order by View Type are added in the second script LayoutOnSheet.

Initially, the sequence of views by type was defined in the script by comparing and filtering views with element id of input values. Then a list of viewports is reconstructed with new sufficient order. (Figure 3.26)

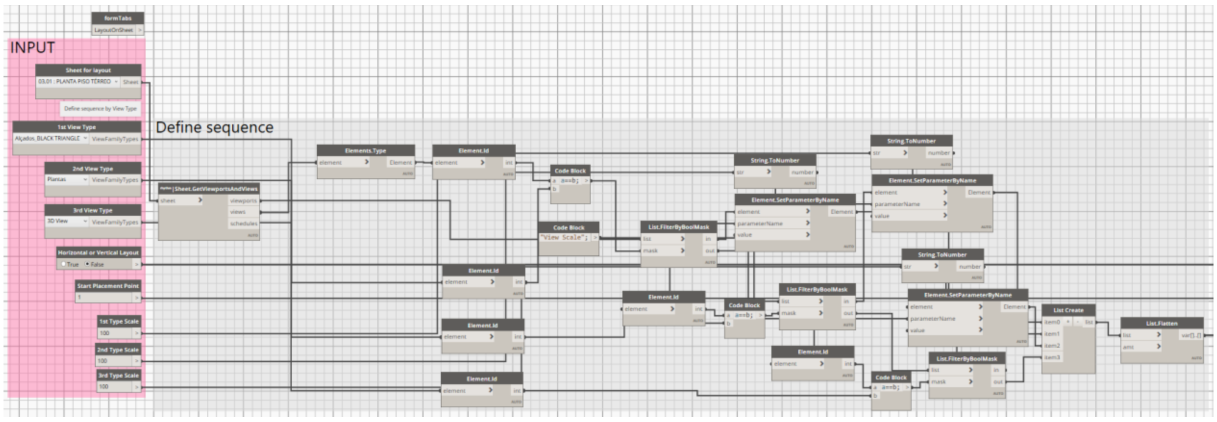


Figure 3.26 – Input Values and Views Sequence definition

After the sequence is defined, dimensions of viewports are extracted through bounding box attributes. Then two sublists are sliced from the list of viewports. The first sublist is the first column, and another sublist is the first row in the assumed layout. After sublists are set up, the largest dimensions in sublists are found and multiplied by the number of columns and rows accordingly to get width and length of container rectangular for layout without overlap. (Figure 3.27) The same definition is performed for Vertical layout but with switching width and length.

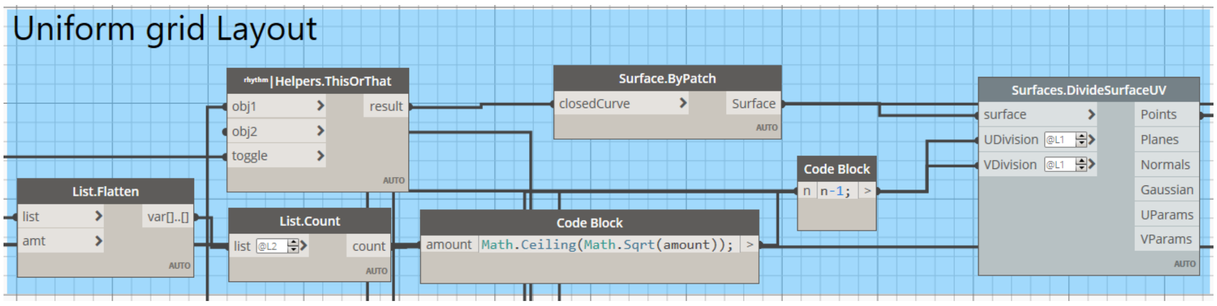


Figure 3.27 – Definition of horizontal Layout Space dimensions

The next step was to divide layout space with grid UVs integers coming from math ceiling round(sqrt(number of viewports)). Thus the grid covers all the viewports locations. (Figure 3.28)

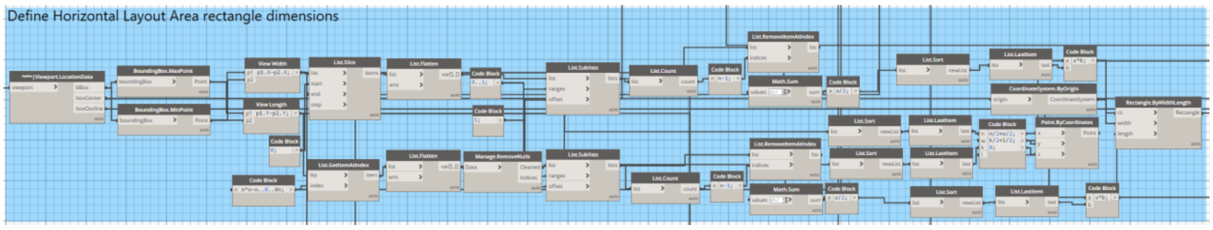


Figure 3.28 – Uniform Grid Layout

A further task was to rearrange a sequence starting from different corner points of layout space in 8 different layouts: four horizontal and four vertical according to the number of start points. (Figure 3.29) So the user can define wished layout at the input. The sorting strategy was applied described by Arturo Tedeschi (Tedeschi, 2014). According to this strategy, points first grouped by X coordinate, then sorted inside the group by Y coordinate, and flattened again.

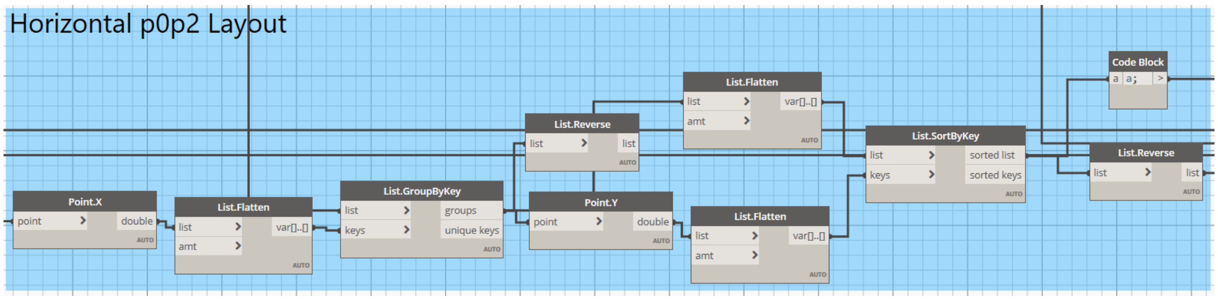


Figure 3.29 – Rearrangement layout definition, repeats 4 times with different connections

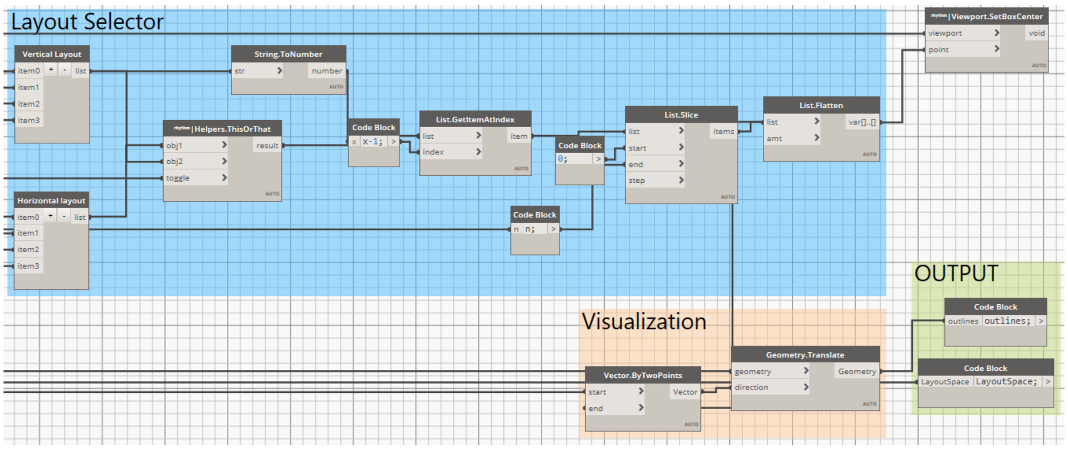


Figure 3.30 – Layout selection definition connecting script with user-defined inputs, translating viewport to correct positions and visualization

Finally, two lists of possible point locations are formed according to Vertical and Horizontal layouts. So, the user first defines the direction of the layout. Then through List.GetItemAtIndex node, the user selects layout defining Start Placement Point from 4 possible positions. In the end, viewports are placed in the correct positions using Rhythm Viewport.SetBoxCenter node and chosen points layout. Translated viewports are visualized in the Dynamo environment. (Figure 3.30)

Finally, all the views are created automatically and put on sheets according to the room asset and just from the room selection. The script applies to multiple selections as well. The workflow described above contains two scripts, one for views generation, and another one is for views layout.

For layout several solutions were found: Miscellany container packer, Refinery Toolkit packer, Pypi Python module Rectangle-Packer (with Python Script). However, after a detailed investigation, it was discovered that all the tools available don’t suit the case implementation. With Dynamo packages such as Miscellany and Refinery Toolkit nodes, it is impossible to constrain the orientation of views in the layout. Python modules such as Rectangle-packer need to be distributed and separately installed on client PC, thus are not appropriate for the case design goal as well.

Consequently, the uniform grid layout is implemented with sequential options and order defined by the user and independent from sheet size. As a matter of fact, the workflow still requires post-processing adjustments. As it was intended by the partner company to put all related views by asset into a single sheet, the amount of generated views and their scales do not necessarily fit in the sheet size. Though it achieves clarity and automatic generation of views sorting them by sheets for the acceleration of documentation authoring. (Figure 3.31)

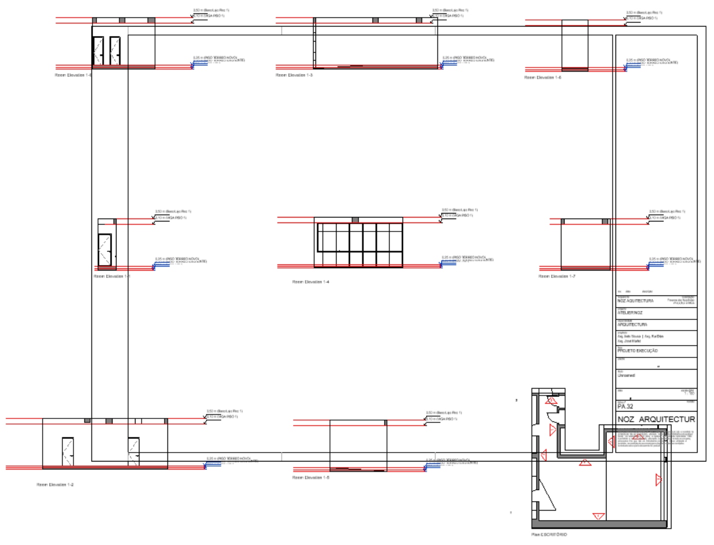
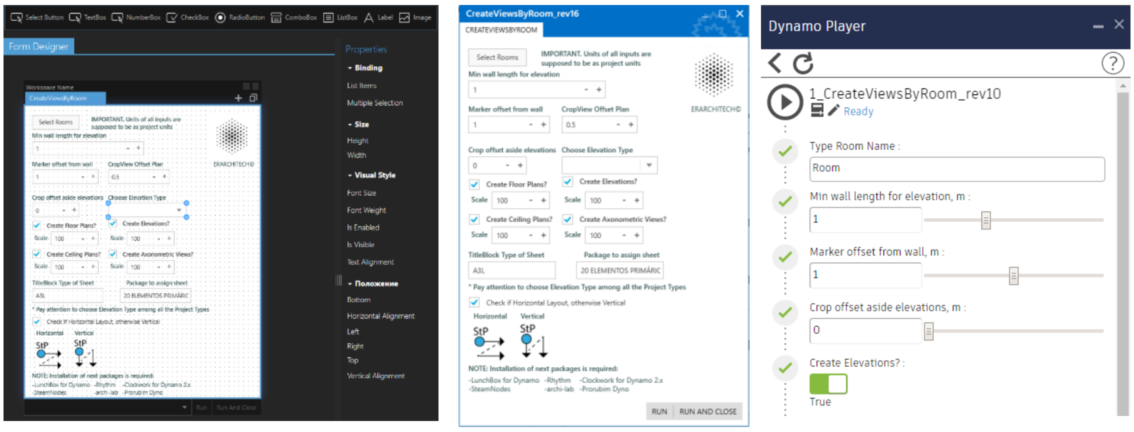


Figure 3.31 – Uniform Grid Layout result

3.2.4. User Interface

The last milestone in this case study but no less important was to elaborate convenient user interface (UI). For this purpose, there is a built-in Revit Dynamo Player. Despite it allows to create UI, it is still very basic and does not provide interface controls such as buttons, their interdependencies (eg. appearing additional parameters when clicking option), inputs presets, popup menus, and descriptions. Alternatively, there are professional platforms for interface creation and managing scripts available on the market, such as Dyno Studio. Such platforms allow developing complete add-ons using visual programming languages. Therefore Dyno Studio platform was used to create and manage UI. The software was provided by Prorubim LLC, Russia. Dyno Studio is the environment that allows us to manage, create, and edit UI for professional Dynamo scripts. The project consists of several stages, each of them gives additional features, which will be integrated gradually. Currently, its development is on the first stage and allows us to create forms and UI for scripts using a simple and clear editor. On the next stage, it is intended to pack scripts and all their dependencies into installators, share, distribute and install.

Consequently, a form was created in Dyno Studio. All the inputs were assigned to appropriate buttons and graphics set up. (Figure 3.32) To connect the script with its interface, node bearing the name of the referred form was added in Dynamo. There can be made two UI. First is the main window that appears separately when running the script in Dyno Player. Another one, Dialog window, pops up when running in Dynamo editor. Unlike the main window relies on user input values only, the dialog window allows to integrate an interface in the script with direct connections with input and output nodes.



Type Room Name : Room
Min wall length for elevation, m : 1
Marker offset from wall, m : 1
Crop offset aside elevations, m : 0
Create Elevations? : True

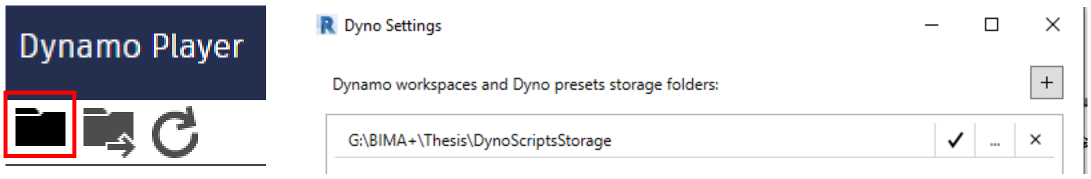
Figure 3.32 - User Interface. Dyno Studio editor (left), Dyno Browser Interface Form (middle), Dynamo Player interface form (right)

MANUAL.

To run scripts, next packages need to be installed from Packages tab in Dynamo editor or website www.dynamopackages.com:

- LunchBox for Dynamo
- SteamNodes
- Rhythm
- archi-lab
- Clockwork for Dynamo 2.x
- Refinery ToolKit
- Prorubim Dyno

Next, to run scripts with the user interface, the installation of Dyno is required. It can be downloaded from http://prorubim.com/en/tools/dyno. It is free but asks to feel the registration form. Otherwise it is still possible to run with default UI from Dynamo Player (Dyno still recommended). In both cases coping all the files in the chosen folder and linking this folder from the player is necessary (Dyno or Dynamo Player). Copy and past archive files into a single default storage folder which can be found on the left upper corner of Dynamo Player or in Dyno settings. Otherwise, create own folder and then link it. (Figure 3.33)



Dynamo workspaces and Dyno presets storage folders:
G:\BIMA+\Thesis\DynoScriptsStorage

Figure 3.33 – Script storage folder in Dynamo Player (left), and Dyno (right)

After installation is complete, scripts can be run. To run CreateViewsByRoom, double click is required on script in Dyno Browser. Interface main window will appear (Figure 3.32 - middle):

- Select room(s)
  *note to push the button "finish selection" on the upper ribbon. To select the rooms for view generating, it is recommended to isolate rooms for selection convenience.
- Fill the constraints with numbers matched to current project units
- Choose the type of view to be created and assign view scale
- Choose elevation Type *Pay attention to choose Elevation Type among all the Project Types available from the pop-up menu
- Define Package Name to assign sheets to
- Choose Horizontal or Vertical Layout (Figure 3.34 – middle, right)
- Run *Be patient waiting script to perform

To run script LayoutOnSheet:

- Choose sheet for the layout from the pop-up menu
- Define order by View Type and assign the view scale
- Choose Horizontal or Vertical Layout (Figure 3.34 – middle, right)
- Choose Start Placement point from 1 to 4 *Points ordered from upper left corner clockwise (Figure 3.34 - left)

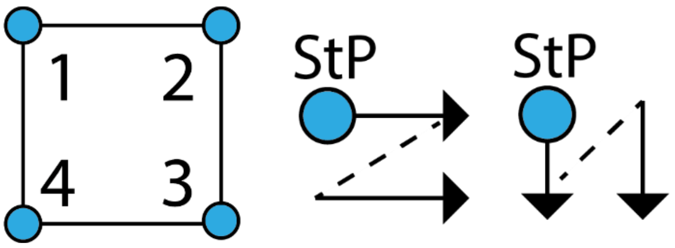


Figure 3.34–diagrams showing input Start Placement Points (left), Horizontal (middle), and Vertical layouts (right)

3.3. CONCLUSION

In summary, visual programming can be an essential part of the AIM methodology in documentation authoring. Among the scope of issues solved by AIM, the most demanding are the generation of frequentative elements, sorting and grouping data, design verification, model data management and exchange, generation of reports and quantity take-offs, and graphical views control.

In general, visual programming is the great assisting means accelerating design documentation authoring process within the first Level of Integration between AAM and BIM environments. (see Introduction chapter, section 1.4) Particularly in this Case Study, scripts are developed in Dynamo GAE for assisting design authoring in Revit.

Within the case scenario proposed by the partner company, automatic views generation for compartments and their organized layout on the sheet are considered. As a result, workflow generating views from a selected room with the automatic layout on a sheet is elaborated. The workflow includes the development of two tools scripted in Dynamo: CreateViewsByRoom, and LayoutOnSheet. The AIM design process is divided into two parts with the creation of corresponding tools for the most agile implementation. It allows conducting changes in layout without the need to recreate all the views and sheets again. Besides, such an approach leads to better performance. Both tools are equipped with the user interface providing convenient usage scenarios to the user without necessary scripting expertise.

Despite discovered advantages of implementation AIM in documentation authoring, some noteworthy issues are identified. These issues detected during the Case Study are related to the packing algorithms problem for view layout. Specific design task requires the adoption of these algorithms under view objects because usually such algorithms are developed for geometry packing. Particularly, all existing packing packages for Dynamo filter exciding views beyond the layout space of the sheet. Therefore, filtered viewports need to be added manually afterwards.

Another packing problem is an absence of the view orientation constraint. Thus, during bin packing, view geometry is automatically rotated on the sheet, which is not allowed. As a result, the uniform grid layout is implemented with sequential options and order defined by the user. Though, as a matter of fact, the workflow still requires post-processing adjustments. Hence, the view packing problem should be further investigated in the future.

All in all, the Case Study has shown the advantages of integrating BIM and AAM tools within a single workflow. Meanwhile, important notes are taken into account for future improvements. In conclusion to the practical study, except methodology and tools investigation, personal achievements include enhancement of scripting skills such as VPL in Dynamo, TPLs (DesignScript and Python), Revit API, and UI creation in Dyno Studio. Moreover, the knowledge is gathered about the scope of existing tools and therefore, related design opportunities and problems. In addition, layout algorithms are learnt.

This page is intentionally left blank

4. CASE 03. FABRICATION

4.1. INTRODUCTION

To materialize the design concept in the real world, human build physical models by different means. We are well-informed about major means throughout the history of architecture. Since ancient times drawing was the ultimate mean for interpretation of concept for construction on-site. It had lasted until the digital revolution when traditional design methodologies facilitated with computer aid in the early 1960s (SketchPad, Ivan Sutherland, 1963). Gradually, CAD (Computer Aid Design) extended with a new batch of technologies such as BIM (Building Information Modeling), AAD (Algorithm-Aided Design), and CAM (Computer-Aided Manufacturing). Thus, together these technologies cover all design stages from conceptual design, engineering, to construction. In addition, HCI (Human-Computer Interaction) technically allows a completely digital framework of the design-to-fabrication process as all the stages have been already digitalized and software used to interpret the model in Object-Oriented Languages. Therefore, tools are implicitly able to communicate and translate data from design to manufacturing automatically.

Consequently, CAM software makes it possible to import and setup the model of BIM component, generating toolpaths and encoding for digital manufacturing and fabrication. Then instructions' file is sent to a machine with a processor onboard which allows to compute them and to control physical operations. Machines in its turn are divided by type. It can be subtractive (e.g. CNC machine - turning, drilling, boring, milling, reaming; EDM - electrical discharge machine; laser cutter; water jet cutter) either additive manufacturing (e.g. SLA -stereolithography; DLP - Digital Light Processing; FDM - Fused Deposition Modeling; SLS - Selective Laser Sintering; SLM - Selective Laser Melting, or fabrication robots such as Kuka.

4.1.1. G-CODE INSTRUCTIONS.

Regardless of the machinery type, communication between CAM software and machines happens through CNC (Computer Numerical Control). Notably, "CNC" title refers now to subtractive machines only. The machine is controlled by computer according to specific input instructions which are delivered as a sequential program such as G-Code (geometric/preparatory code) and M-Code (miscellaneous code). Whilst M-Code relates to executing general functions such as tool, spindle speed, and coolant changes, G-Code refers to specific machining movements called cycles used for making geometry profiles. Both programs are generated automatically from toolpaths setup in CAM software e.g. Autodesk Fusion 360.

To understand better how G-code controls machine during manufacturing, it is worth to consider its general structure. G-code follows some variation of the alphanumeric pattern:

N## G## X## Y## Z## F## S## T## M##

N: Line number
G: Motion
X: Horizontal position
Y: Vertical position
Z: Depth
F: Feed rate
S: Spindle speed
T: Tool selection
M: Miscellaneous functions

Alpha numeric codes are used for programming as they are a simple way to:

1. Define motion and function (G##)

2. Declare a position (X## Y## Z##)

3. Set a value (F## and/or S##)

4. Select an item (T##)

5. Switch something on and off (M##), such as coolant, spindles, indexing motion, axes locks, etc.

For instance,

G01 X1 Y1 F20 T01 M03 S500

would generally indicate a linear feed move (G01) to the given XY position at a feed rate of 20. It is using Tool 1, and the spindle speed is 500. Miscellaneous functions will vary from machine to machine, so to know what the M-Code means, the machine's instruction manual will need to be referenced. Thus, instructions are still highly dependent on a specific machine. Although all machines read the common structure of instructions, the M-Code part stays proprietary. It means that the machine used for manufacturing shall be in the reference list of CAM software.

As seen from above, the workflow BIM to manufacturing requires involving third-party tools for interpreting model and creating toolpaths specific to manufacturing machine generating G-Code instructions. It complicates the workflow.

However there some exceptions when vendors provide their tools integrated into BIM authoring software. For instance, such a tool named Kuka/PRC (Parametric Robot Control) is provided by the Association for Robots in Architecture. The tool builds upon the accessible visual programming systems including Grasshopper and Dynamo. Kuka/PRC allows programming the robots directly from BIM or CAD platform with accurate simulation and optimization abilities. So It helps to quickly verify the robot program and ensure that there are no collisions or unreachable points. With the immediate feedback, the designer can intuitively solve problems by interacting with parametric definition and observing the results. Although, it is still a proprietary tool made exclusively for Kuka. (Figures 4.1, 4.2)

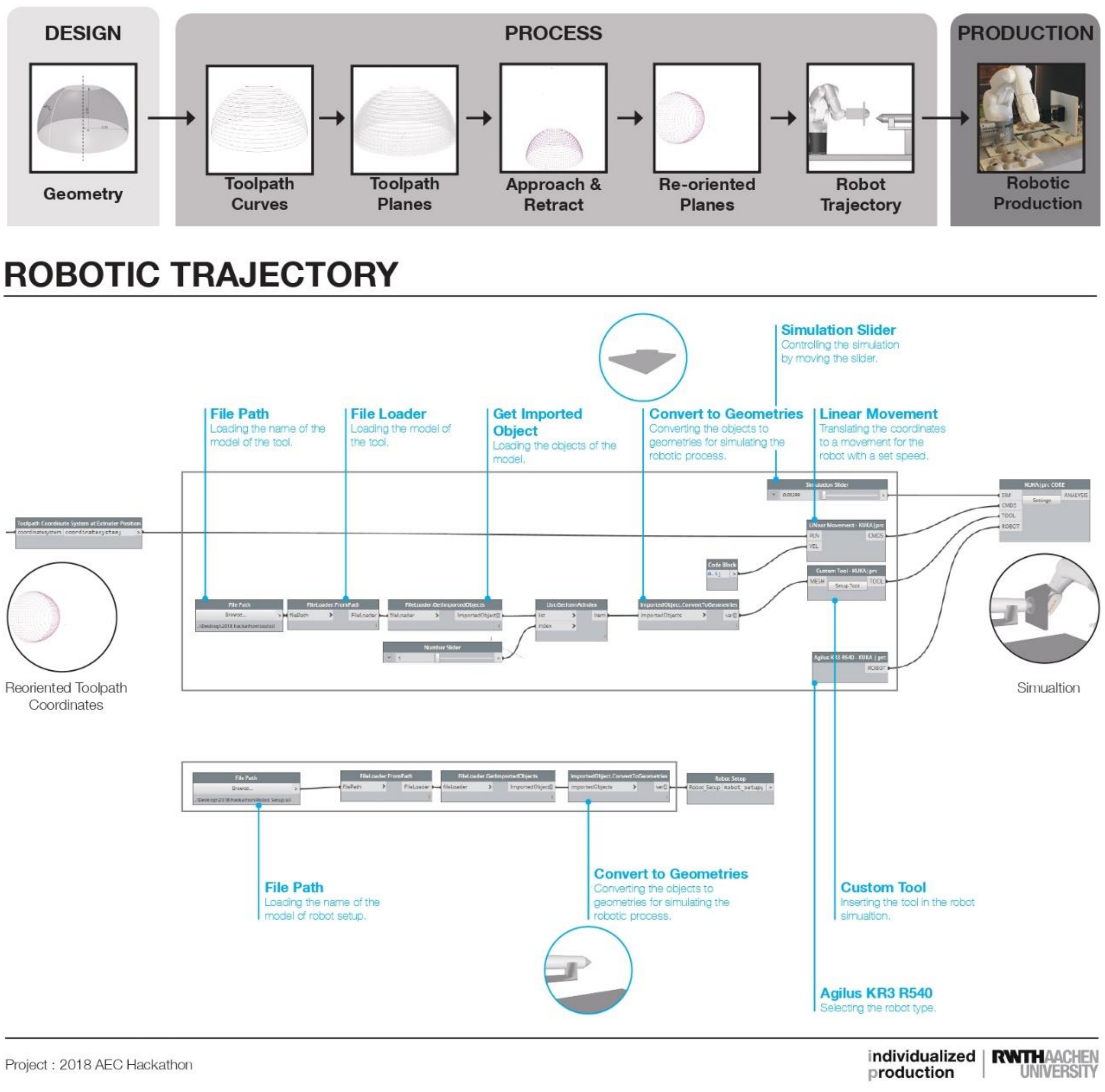

DESIGN
Geometry
PROCESS
Toolpath Curves
Toolpath Planes
Approach & Retract
Re-oriented Planes
Robot Trajectory
PRODUCTION
Robotic Production
ROBOTIC TRAJECTORY
Simulation Slider
Controlling the simulation by moving the slider.
File Path
Loading the name of the model of the tool.
File Loader
Loading the model of the tool.
Get Imported Object
Loading the objects of the model.
Convert to Geometries
Converting the objects to geometries for simulating the robotic process.
Linear Movement
Translating the coordinates to a movement for the robot with a set speed.
Reoriented Toolpath Coordinates
Simualtion
File Path
Loading the name of the model of robot setup.
Convert to Geometries
Converting the objects to geometries for simulating the robotic process.
Custom Tool
Inserting the tool in the robot simulation.
Agilus KR3 R540
Selecting the robot type.
Project : 2018 AEC Hackathon
individualized production
RWTHAACHEN UNIVERSITY

Figure 4.1 – Kuka/PRC workflow (top), Robotic trajectory scripted in Dynamo (bottom)

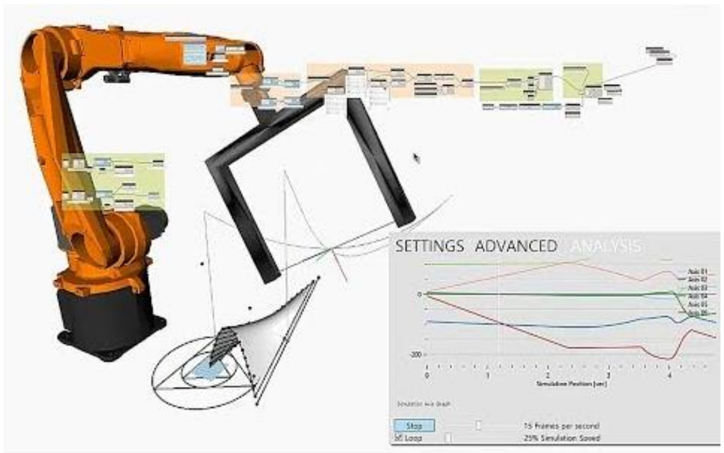


Figure 4.2 – Programming Kuka robot in Dynamo test

4.1.2. Model accuracy and reliability issues from a manufacturing perspective.

Another issue with digital fabrication is the accuracy and reliability of the BIM model. Despite BIM implementation and its culture grows rapidly, models transferred for manufacturing mostly safer with lack of usable data. The main reason for this is the lack of focus. Building information models are typically developed from front to back, driven by project development and design coordination but with limited knowledge about the fabrication environment. Without a clearly-defined objective, almost all planning models miss the target of being usable for production. Instead, fabricators start remodelling from scratch when they finally become involved in the process. The phenomenon of dumping and remodelling information at every process stage has been described by Borrmann et al. (2015).

Practical experience shows that digital information handed-down from previous planning stages is too detailed and unreliable. Since it is hard to determine which parts of ‘shotgun models’ are trustworthy, the pragmatic and safe approach is to throw the input away and rebuild altogether.

4.1.3. Concept of Design for Manufacturing and Assembly in BIM.

To curtail the influence of the factors described above in a prefabrication project, every design decision needs to be checked against manufacture and assembly, ideally in an automated fashion. I consider vitally important BIM manufacturing innovation is the development of a materially-informed digital design methodology that could be used to predict and tune the final shape and translate a design geometry to the material information required for production. Thus, to be effective, the predictive model must be accurate using material input parameters within ranges that can be collected and implemented in an industrial context.

In other words, design needs to take fabrication and assembly seriously at an early stage of the process, to avoid unexpected delays and costs later on. In product design, this is called “Design for Manufacture and Assembly” (DfMA) and has been an established field of research since the late 1980s (Andreasen,

1988) In the AEC industry up to date, these questions are implicitly resolved by the fabricator at the engineering stage of a project, with no chance to optimize the design. It implies the only one-directional workflow where the original concept loses its quality along the way of design transition toward manufacturing. As a result, both the designer and client are not satisfied as the costs and idea suffer in consequence. To resolve transition issues in future industrialized BIM processes, DfMA needs to be applied methodically at early stages, because the development of prefabricated components is, in fact, product design as well. The main inferred principles of DfMA (ICE, 2020) are:

- Minimise the number of components: Thereby reducing assembly and ordering costs, reducing work-in-process, and simplifying automation.

- Design for ease of part-fabrication: the geometry of parts is simplified and unnecessary features are avoided.

- Tolerances of parts: part should be designed to be within process capability.

- Clarity: components should be designed so they can only be assembled one way.

- Minimise the use of flexible components: Parts made of rubber, gaskets, cables and so on, should be limited as handling and ASSEMBLY is generally more difficult.

- DESIGN for ease of assembly: for example, the use of snap-fits and adhesive bonding rather than threaded fasteners such as nuts and bolts. where possible a product should be designed with a base component for locating other components quickly and accurately.

- Eliminate or reduce required adjustments: designing adjustments into a product means there are more opportunities for out-of-adjustment conditions to arise.

A different workflow has been developed by Fabian Scheurer and Hanno Stehling after more than a decade of experience in freeform timber projects and continuous discussions on the aforementioned modelling conundrums. (Figure 4.3) All topics, from the abstract context to concrete fabrication, assembly and maintenance, are addressed from the outset in an ‘agile’ fashion to provide a viable solution at every stage. This requires enough knowledge about production to be brought upstream to inform the design regarding smart integrated opportunities, either by tapping into the know-how of production specialists at early project stages or by developing and following common DfMA guidelines for prefabrication. To efficiently arrive at a high-quality model, the volume of information needs to be kept to a minimum. Only the decisions made and justified at any given stage need to be reflected in the data, thus the process ‘re-surfaces’ to the appropriate level after each round. In addition, the ‘abstract’ models are not replaced by the more ‘detailed’ model at the next stage, but instead are kept alive and extended by the additional information. This requires updates to be made to the abstract model in conjunction with later findings, while therefore the probability of change at later stages should also be addressed before adding to any model. (Stehling, 2020)

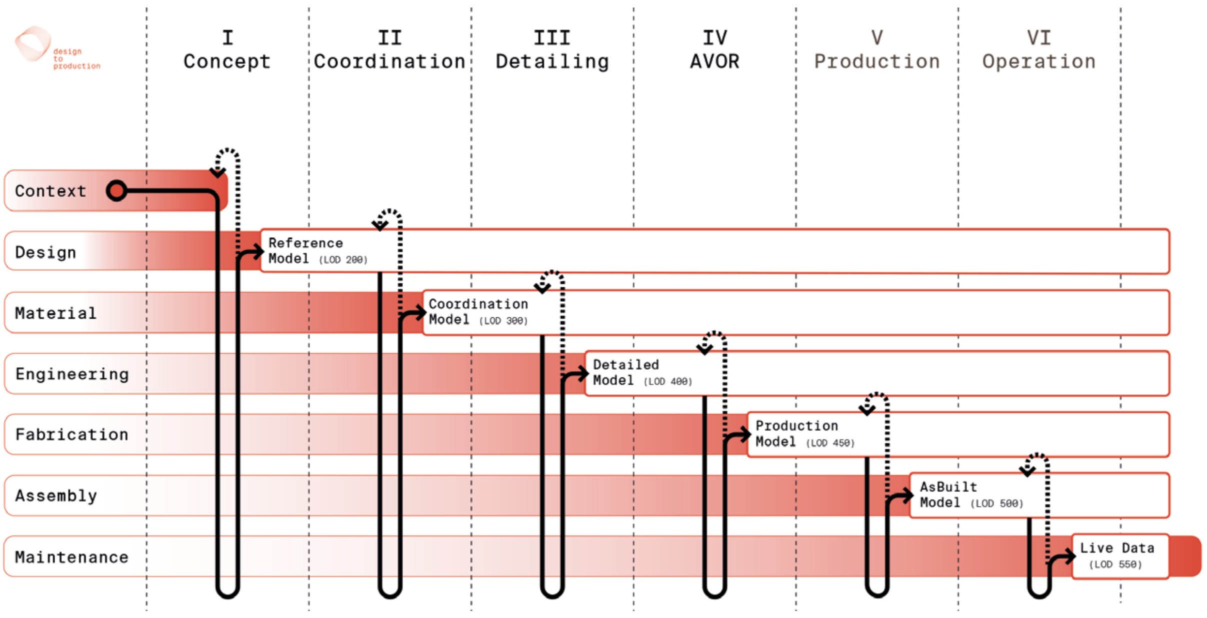


Figure 4.3 - Agile Design-to-Production process model. The diagram shows a planning process through different stages in time (from left to right) and varying abstraction levels (from top to bottom). Refered from book "Design Transactions"

4.1.4. Advantages of Parametric model for DfMA.

Typical accuracy for CNC manufacturing should be within the range of 0.5mm and is necessary to fabricate "watertight" fitting elements, e.g. slotted plates fixed to timber parts by steel dowels.(Stehling, 2020) A digital model that is to control a CNC machine directly must be at least as accurate as this, but expecting this level of precision at an early stage of the design process is unrealistic. Uncertainty and change will always be part of every planning process, resulting in multiple iterations. In this context, models cannot be efficiently created with conventional 'manual' methods, but rather need to be generated based on parametric rules.

Consequently, the parametric model becomes a new agile method based on encoding the generative scheme establishing these rules. This method allows the model to stay flexible and adapt to changing parameter values but, at the same time, prove highly accurate after each update. Creating such models requires a high degree of systematization to untangle and prioritize dependencies; therefore, decisions need to be made about the underlying rules and structures. On the upside, due to this systematic approach, parametric models are not only accurate but are also reliable. The validity of the results depends more on the rules and inputs than the capabilities of the individual modeller. Debugging of the parametric model is much easier as well. Farther, some cases are provided implementing the concept of DfMA and parametric model for manufacturing in different scales.

4.1.5. Case 01: BioMat Pavilion 2018, Segmented Shell of Biocomposites and Wood, University of Stuttgart.

The selected design was based on a complex 3D textilelike structure, chosen to represent new innovative design but mainly for structural reasons. The structure was composed of two interconnected layers that provide the spatial rigidity which was found necessary for the used material. For such a complex structure, a parametric model was necessary. The structure of the pavilion consisted of 360 segments. The structure was designed according to size limits of flexible boards, mechanical properties and building site limitations. The model was also used for smart numbering, structural optimization and fabrication control. Through optimization, only four moulds for vacuum-assisted fabrication were used to produce the 360 different segments. The design was constantly updated by feedback from material tests and structural calculations provided through international cooperation with colleagues from the Technical University Eindhoven and KE institute of the University of Stuttgart.

Off-site fabrication of large free-form biocomposite segments was mass-produced by the architectural students through a digital fabrication setup of CNC milling and closed moulding using a vacuum appliance – of around 360 segments. Later, these items were assembled to form 120 larger individual elements, each composed of 3 segments. The on-site assembly started by erecting three supporting laminated arches positioned precisely according to the 3D scan model of the site made in cooperation with a geodesic team from IIGS Institute at the University of Stuttgart. Successive 3D scanning of the site before, throughout and after erection enabled guaranteed high precision of the arches to reach the accuracy for assembling the 120 biocomposite elements. The elements were first interconnected through plywood plates and bolts into four triangular groups on the ground, then lifted on to the beams and fixed in position as shown in figure 3. After completion, repetitive 3D scanning took place and was compared with the parametric model (Figure 4.4) The differences between the digital model and after erection were in the allowed tolerance range.

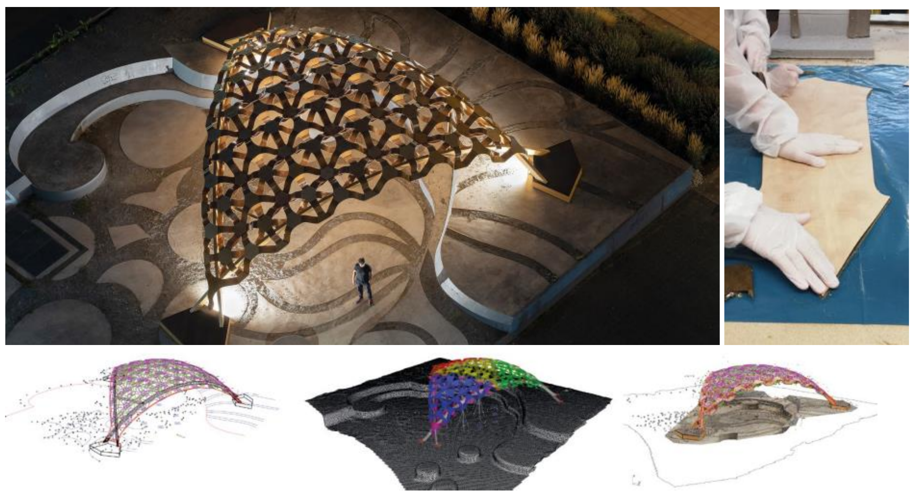


Figure 4.4 – (1) Biomat Research Pavilionafter completion. (2)Off-site fabricationof individual segments.(3)Parametric model. (4) 3D scanned building site. (5) Comparison between 3D scanned completed pavilion and parametric model. All photos belong to BioMat at ITKE/University of Stuttgart

4.1.6. Case 02: Kuwait International Airport, AIM System

In order to increase the capacity of Kuwait International Airport, a new state-of-the-art terminal 2 building designed by Foster+ Partners is currently under construction and is due for completion in August 2022. The project is a turning point with regard to the large-scale use of wide-span reinforced concrete shells. As described in the present case, the extensive use of prefabrication and design-to-fabrication processes made it possible to build geometrically complex reinforced concrete shells without complex formwork or expensive scaffolding systems. (NIERI, 2020)

Given the sheer scale of the project and the geometrical variation of the different components, the authors invested a considerable amount of time and energy in the development of a semi-automatised process. Starting from a parametric geometrical model, this process allowed for structural calculations of the different components and led to modular detailing driven by forces and geometry. This was a particular challenge for computers and human beings alike. Model size and complexity were pushed to the limit of what current FE and BIM models can handle within the building construction field. Each task needed to be first parametrically defined, then automatized and finally optimized. For this reason, the scope of the engineer was not only to design and develop individual structural elements. The task was rather the definition of methods and rules that could be applied to large groups of elements presenting common characteristics (like topology, function or geometry). The result was a dual digital model containing the information for fabrication and for the structural analysis utilizing AIM approach.

The “Central Data Model” containing all the important information (dimensions, material, etc.) of each component (beams, cables, connections, etc.) was based on the software McNeel Rhinoceros. Within this platform almost all the elements were defined by scripting (C++, C# and Grasshopper), using specific interfaces (some of which were developed in-house by the authors). This unique source was employed to share information with other software programs (FEM, BIM, CAD, etc.) and to develop sub-models.

Such an approach was particularly challenging in the initial part of the project. It required breaking down the complexity and defining the relationship between geometrical and structural parameters. This was an essential prerequisite in order to set up the correct workflow. It was not always easy for the client to understand why this process required a large amount of time and needed to be well prepared. Thanks to this approach, late changes – which may always happen, for example due to difficulties with regard to logistics or procurement, special wishes by the client, etc.) – could be addressed with a reasonable effort. The authors thus overcame one of the limits of traditional approaches used in the past. Furthermore, it was possible to define a fully coordinated package constituted of structural models (SOFiSTik, and custom structural design tools), BIM coordination models (Revit, Rhinoceros, Navisworks, etc.), BIM models for production (Tekla, Rhinoceros, Allplan, etc.) and CAD drawings (overview drawings, formwork drawings, rebar drawings, etc.).

The tools that were developed do not only allow for a high degree of prefabrication within the tight frame defined by budget and time, they also allow for a parametric variation of the different bays. Moreover, the development and integration of automatized processes and a direct chain of control from design to fabrication were fundamental to guarantee the high level of precision required.

Figure 4.5 – Kuwait International Airport Terminal 2, Foster+Partners.

From the aforementioned cases implementing DfMA concept and parametric modelling, advantages of such an approach in design-to-production workflow are seen. In both cases models are material-informed from the early stage of the design and highly parametrized. Taking into account that both scenarios underwent plenty of revisions before final assembling, DfMA methodology allowed to keep consistent workflow during all design stages just varying values and optimizing model' performance with reasonable efforts. Though it still requires special knowledge about materials and fabrication as well as programming skills to create a parametric model and automatize the design-to-production transition, as there is no ready enterprise solution yet on the market.

4.2. CASE STUDY. PANELIZATION TOOL

4.2.1. Case Background

In the present Case Study, a model-to-fabrication workflow is considered based on the existing design of furniture within the interior project proposal of NOZ Arquitectura.

A proposed design requires the cutting of sheet material, such as stone sheets and ply panels to be assembled on-site, and for joinery. Since at NOZ Arquitectura, designers often include in proposals specific joinery items, they have to tackle closely with production. Though many carpentries are still low-tech and do not have the resources to plan an optimized sheet layout. As a result, designers shall provide Panelization drawings of the proposed items themselves. Currently, the workflow is completely manual and takes a lot of time preparing the model. Thus, the company seeking for improvement to gain automation and optimization in manufacturing of their design products.

OBJECTIVES

Consequently, responding to the existing challenge in the company, the next objectives were defined:

1. Automatize the panelization of existing BIM elements directly from BIM authoring platform to CAM.
2. Enable effective setup of inputs such as sheet size, grain orientation, and the gap distance between neighbour elements in the layout.
3. Enable different layout options depending on desired manufacturing results:

• **best fit.** The algorithm should lay the various pieces in order to make the best use of the sheet area, regardless of sheet grain orientation.

• **grain oriented**. The algorithm should lay the various pieces aligned to the desired grain orientation of a sheet. This can be the stones veins or wood grain of a ply sheet, etc.

• **cut-pass style.** Can be single or double. Depending on if two pieces are placed back-to-back and split with a single path or if they are distanced and each piece has own cut profile. If “double cut-pass style” is selected, the distance should be defined allowed between cuts of different parts.

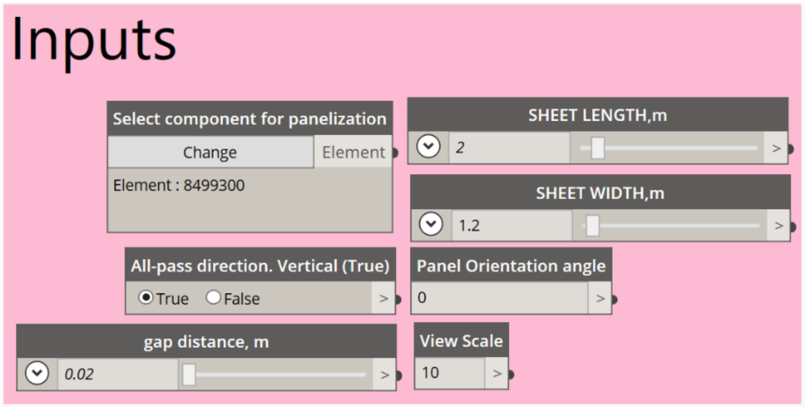
WORKFLOW

The workflow includes a single script definition that planifies the Revit component into panels and lays out them according to sheet size and grain orientation selected. The definition contains four milestones:

1. Panel geometry extraction
2. Panel size verification
3. Packing panels on sheets and rebuilding Guide Model
4. Generation of drawing

4.2.2. 1st Milestone: Panelization and Mapping

Before fabrication, it is required to define manufacturing rules which will affect the layout. Such rules are identified in collaboration with the partner company. The first rule taken into consideration is all-pass direction. So, vertical or horizontal parts are considered as all-pass elements. Consequently, other panels are getting cut by selected all-pass elements. Another defined rule is the grain orientation of panels. It is supposed to be aligned with the grain orientation on the sheet to get desirable results. So, to define this rule, the panel orientation angle is added to inputs. The third rule is gap distance. Regarding "double cut-pass style", it defines the distance to allow between cuts of neighbour panels in the layout. Regarding defined rules, the next inputs are determined: selector for panelized component selection, sheet dimensions, gap distance, all-pass direction, view scale, and panel orientation angle. (Figure 4.6)


Inputs

Select component for panelization: Change | Element — Element : 8499300

SHEET LENGTH,m: 2

SHEET WIDTH,m: 1.2

All-pass direction. Vertical (True): [✓ True] False

Panel Orientation angle: 0

gap distance, m: 0.02

View Scale: 10

Figure 4.6 – Input parameters of Panelization script definition

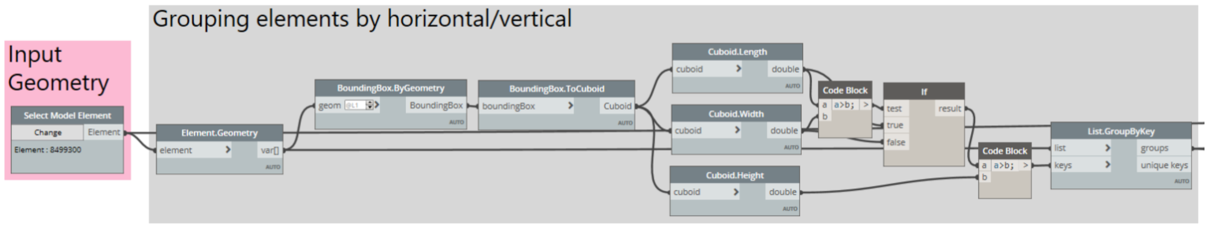
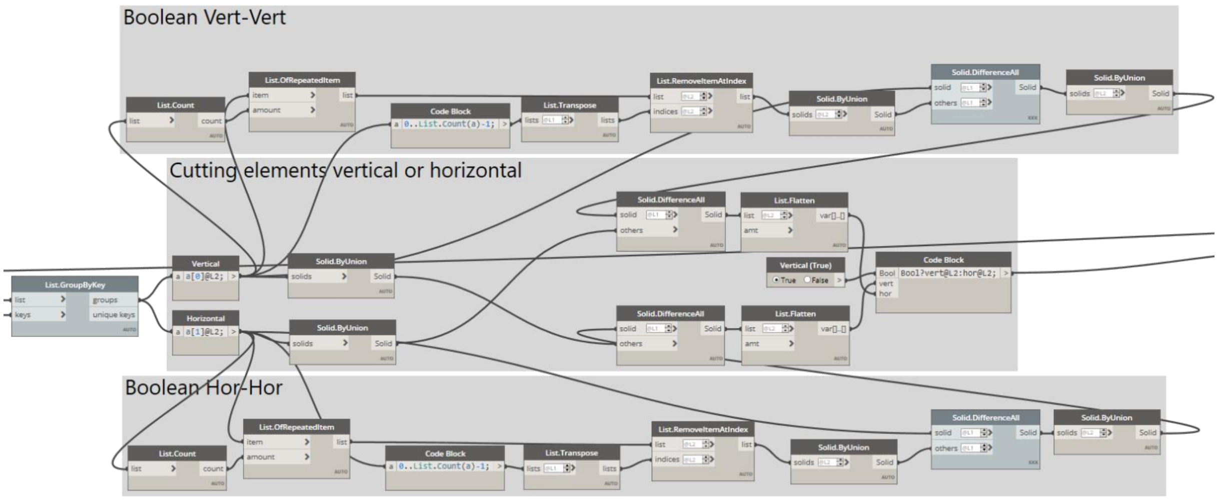


Figure 4.7 – Grouping sub-elements of selected component by orientation of bounding box (Dynamo)

After all the inputs are defined, the geometry can be selected for panelization. Once gathering the geometry, the first step is to group its sub-elements according to their orientation for further all-pass cutting rule implementation. To perform this task, BundingBox.ByGeometry and BoundingBox.ToCuboid nodes are used to extract elements dimensions through cuboid properties such as Cuboid.Length, Cuboid.Width, Cuboid.Height. Next, these properties are compared and the largest value is found. Since only two possible directions are considered, Boolean values resulting comparison become keys for grouping elements into horizontal and vertical groups. The all-pass direction is controlled by input Boolean value. (Figure 4.7)

The second part of the script is cutting panels by all-pass elements. Initially, Revit component' sub-elements grouped by direction can still intersect with each other because of inaccurate modelling. To prevent these intersections for manufacturing, first, substruction operations are performed within groups. (Figure 4.8, Boolean Vert-Vert, Boolean Hor-Hor) A further step is to cut the rest of the panels by all-pass elements. For cutting Solid.DifferenceAll node is applied. All-pass elements, horizontal or vertical, are defined by input Boolean coming through "If" condition selector. (Figure 4.8, cutting elements vertical or horizontal). The SOS (refer to Glossary) is presented in Figure 4.9.



```
0..List.Count(a)-1;
a[0]@L2;
a[1]@L2;
Bool?vert@L2:hor@L2;
```

Figure 4.8 - Cutting panels with all-pass elements

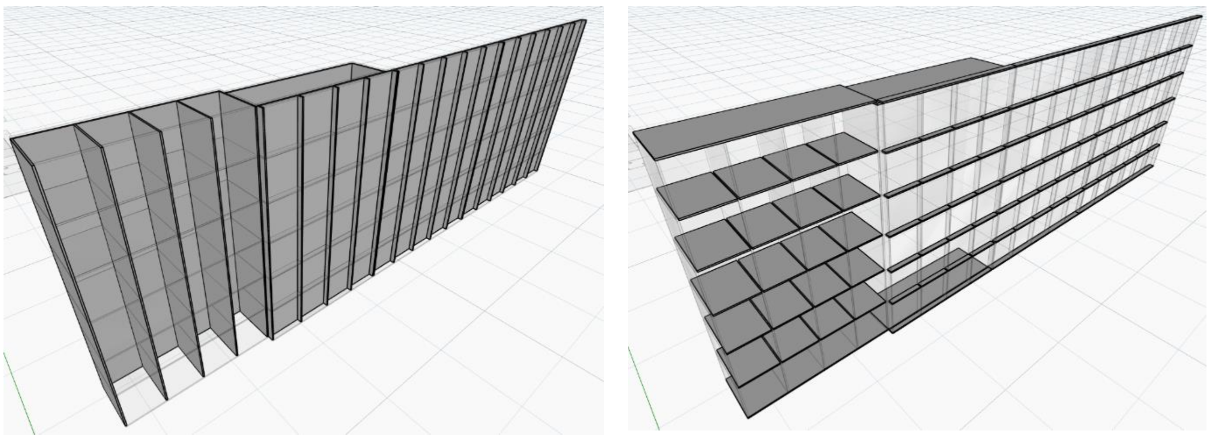


Figure 4.9 – Revit component divided on vertical and horizontal parts. Vertical elements are all-pass (Dynamo)

After cutting the elements, the next step is to explode substracted sub-elements to separate items and map their surfaces on XY plane for further layout. Accordingly, sub-elements are exploded by converting sub-elements to polysurfaces and then extracting solids out of these polysurfaces with node Polysurface.ExtractSolids. (Figure 4.10, Extracting Separate Items) After extraction of all the items corresponding panel elements, their projections are gathered through exploding solids again to surfaces, sorting them by area, and finally selecting the surface with the largest area which is considered as a panel projection. (Fgure 4.10, Extracting Panel Contours) Eventually, panel surfaces are mapped in the linear array on the XY plane. To do so, point array in XY plane is created matching the number of panels, and surfaces are transformed from the source coordinate system to coordinate system originated in created points. (Figure 4.9, Map Panels to XY)

As a result of the first milestone, we have panel surfaces mapped on the XY plane and prepared for packing them on sheets. (Figure 4.11)

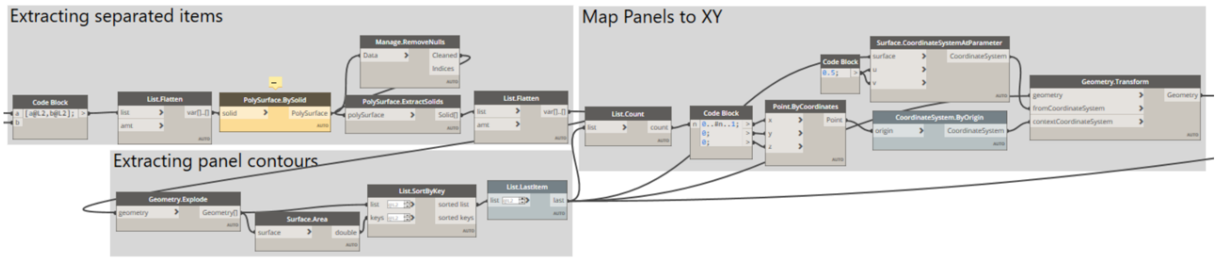


Figure 4.10 - Exploding component to separate items with extracting its contours and mapping them in XY plane (Dynamo)

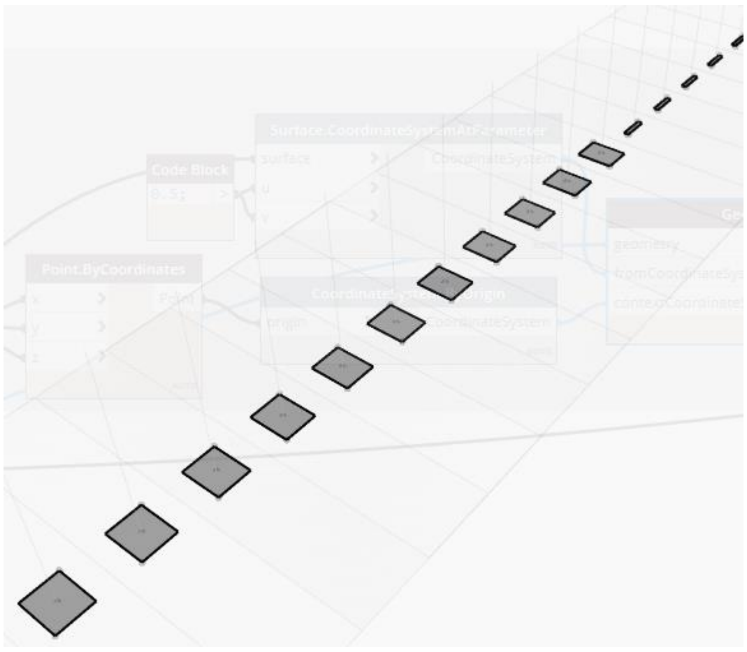


Figure 4.11 - Mapping panel contours to XY plane for further layout

4.2.3. 2nd Milestone: Panel Varification and Splitting

To pack panels on the sheet, first of all, the panels shall be verified for fitting the sheet size. Consequently, the second part of the script definition is dedicated to panel size verification. The panels are verified if they exceed the size of the sheet. If panels are oversized, they divided into two equal parts along the longest dimension. The process is recursive loop while the panel remains larger than sheet dimensions.

The loop body contains an internal definition. First, panel dimensions are extracted through its bounding box extreme points coordinates subtraction. Then the maximum and minimum dimensions for each panel are determined. (Figure 4.12) The reason why the bounding box is used for dimensions extraction instead of panel contours is that panels are not necessarily rectangular geometry and may have more than four sides.

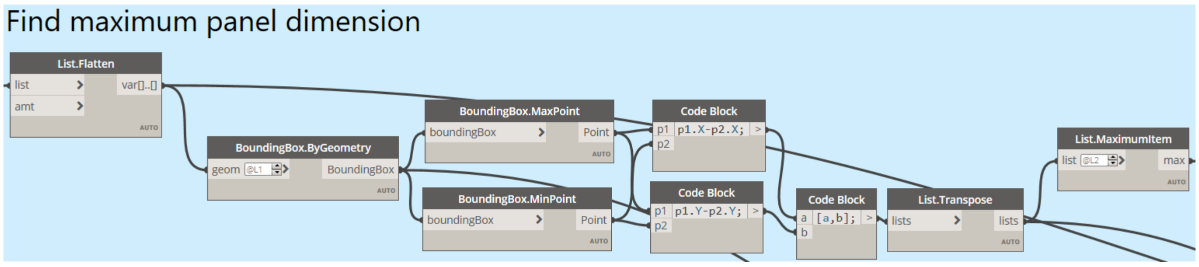


Figure 4.12 - Extracting panel dimensions and its maximum values

Secondly, panel dimensions are compared with input dimensions of manufacturing sheet. Minimum and maximum values are compared accordingly. Then math operator “&&” is applied for logical conjunction of boolean values of both dimensions comparison making general conclusion if the panel exceeds sheet space. The output of “&&” node is further used as a mask filtering panels if they exceed sheet sizes. (Figure 4.13)

Lastly, exceeded panels are split by the splitting line on the base of their bounding boxes. Bounding boxes are converted to rectangles based on its dimensions. Then bounding box rectangles are filtered with the same rule as panel surfaces. Once rectangles are filtered, the outline curves are extracted, sorted by length, and the largest two items in the sorted lists are gathered. Finally, the splitting lines are created between the middle points of the longest line pairs with the node Curve.PointAtParameter. In the end of loop body, exceeded panels are split by these splitting lines with the node Geometry.Split (Figure 4.14)

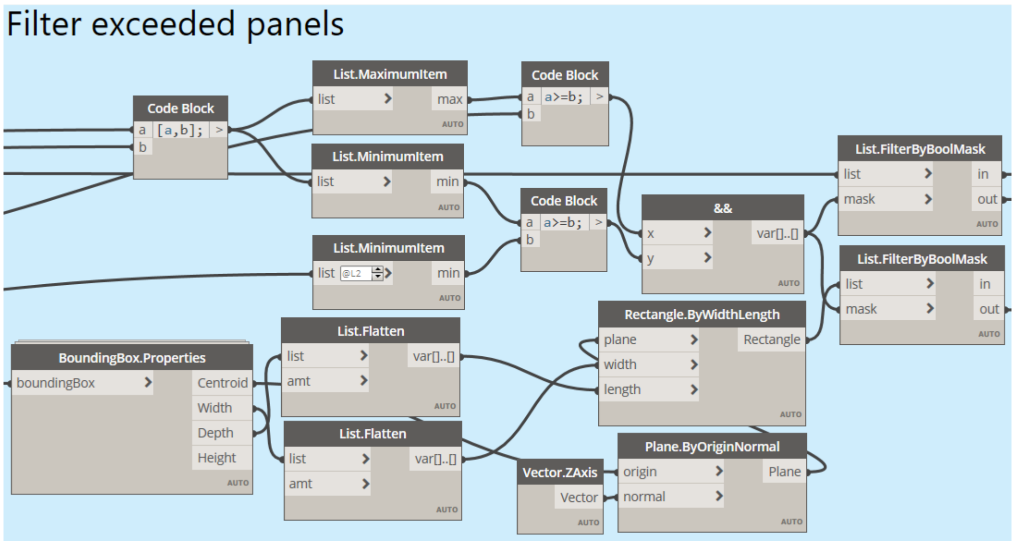


Figure 4.13 - Filtering exceeded panels through its dimensions comparison

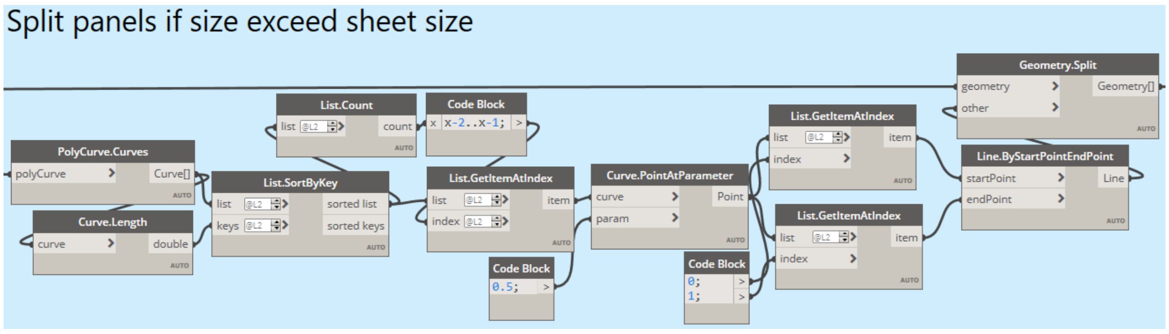
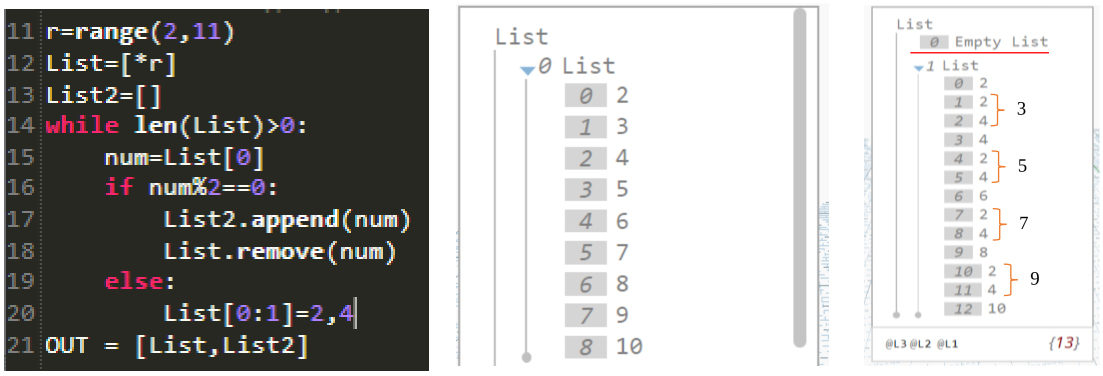


Figure 4.14 - Spliting exceeded panels

After script definition for splitting exceeded panels is written, loop shall be applied to downstream operation into the deeper levels. By this way, parts of split panels on the first level, which still exceed of sheet size, will be split on the next level until all the panels fit sheets. The loop can be defined through the Python custom node inside the Dynamo script definition.

Before writing a Python script application for splitting panels, the application simplification is necessary to understand the algorithm and testify it for the simple case. Thus, the initial root algorithm is identified as interpolating elements in a list based on test pass. So, if the test is failed, the element is divided into two subelements. Otherwise, the element is written into the output list. This operation is looped until all the elements are satisfactory to the test.

As a sample case, a range of numbers is used for identical modifications as for the splitting list of panels. A conditional statement is passed if the number is even otherwise it is substituted by two new even numbers in the list. If the test is passed, the number is translated to the output list and at the same time, it is removed from the initial list. “While” loop is applied with the initial list positive length test. So, the operation iteration lasts while the length of the list is positive. (Figure 4.15, left) In other words, the loop is active until all the numbers are removed from the initial list. As a result of the sample code, the initial list and output lists are demonstrated. (Figure 4.16, right) Finally, the algorithm works properly and is ready to be projected on a more complex task of splitting panels.



```
r=range(2,11)
List=[*r]
List2=[]
while len(List)>0:
    num=List[0]
    if num%2==0:
        List2.append(num)
        List.remove(num)
    else:
        List[0:1]=2,4
OUT = [List,List2]
```

```
List
 0 List
   0  2
   1  3
   2  4
   3  5
   4  6
   5  7
   6  8
   7  9
   8  10
```

```
List
 0 Empty List
 1 List
   0  2
   1  2 }
   2  4 }  3
   3  4
   4  2 }
   5  4 }  5
   6  6
   7  2 }
   8  4 }  7
   9  8
   10 2 }
   11 4 }  9
   12 10
@L3 @L2 @L1                {13}
```

Figure 4.15 - Sample loop code. Python code (left), initial list (middle), output lists (right) –the initial list becomes empty

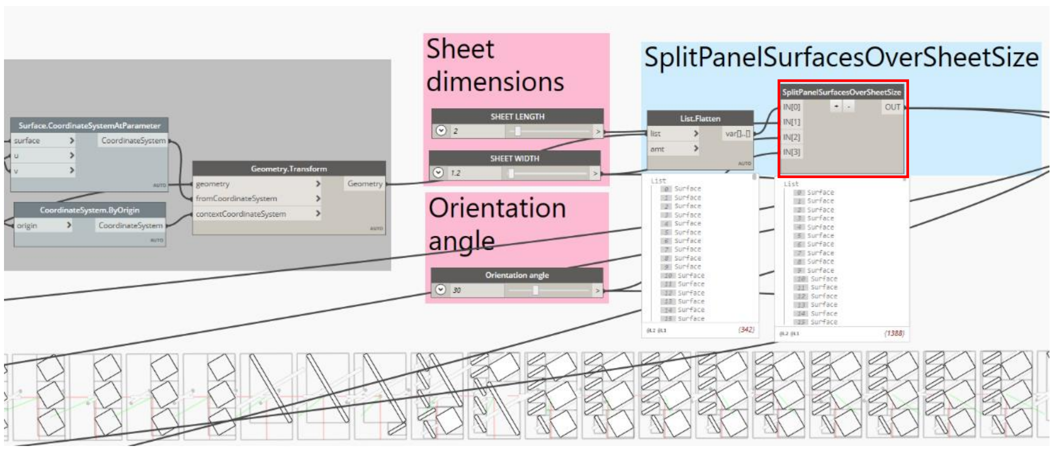

Sheet dimensions
SHEET LENGTH
SHEET WIDTH
Orientation angle
Orientation angle
SplitPanelSurfacesOverSheetSize
Surface.CoordinateSystemAtParameter
CoordinateSystem.ByOrigin
Geometry.Transform
List.Flatten
SplitPanelSurfacesOverSheetSize

Figure 4.16 - Script part using Python definition (in red frame) for splitting panels

Splitting operation is written in Python node inside the Dynamo script according to simplified numerical analogue represented above. Inputs for Python node are the next:

- Panel surfaces (Panels)
- Sheet dimensions (dim1, dim2)
- Orientation angle (Degree).

The logic inside the loop mainly inherits the sample code described above and the definition written previously as the graph in Dynamo. However significant difference from the former definition is the added panel orientation feature for grain direction alignment. (Figure 4.16) Further code description is structured with a top-down approach.

First of all, a splitting strategy and code structure are well defined. The “while” loop is applied as code core to iterate over panels. For splitting panels, the test shall be passed comparing dimensions between panels and sheet. To start with, sheet maximum and minimum dimensions are determined. (Figure 4.18, lines 64-65) The dimensions of panels are then extracted within the loop as instance operation. Consequently, following the code structure of the sample (Figure 4.15), the second list is initiated as output which will collect all the panels after splitting operation. (Figure 4.18, Panels2 – line 67) Next, “while” loop is used with a test for the positive length of input “Panels” list. So, the loop will last until the length of the “Panels” list will become zero. Hence, all panels will pass the loop and get split if necessary, getting stored in the output list. To be more specific, after each iteration, if the panel fits the sheet, it goes to the output “Panel2” list and gets removed from the initial “Panels” list. If the panel does not fit the sheet, it gets divided and replaced with smaller parts in “Panels” list. Sequentially, the operation goes farther downstream. Therefore, the length of the “Panels” list varies with every iteration. Zero value of length will indicate the end of “while” loop execution. (Figure 4.18, “while” loop – lines 70-114)

The last but not the least part of the splitting strategy is grouping divided panels by their inheritance from input panels. So, the groups should match with input panels. This procedure allows farther proper rotation and reconstruction of panels in 3d space. To do so, the second loop “while” is added inside the first one with the same logic. (Figure 4.18, internal “while” loop, lines 75-112) It is applied individually for each panel from input “Panels” list and collects split elements first in internal “PGr” list (PanelGroup) for the initial panel until it will fully fit the sheet. (Figure 4.18, PGr.append – line 111) Once all parts of the panel fit, the internal loop is closed. PGr list is added to Panels2 list and the main loop is repeated for the next panel again unless the length of the input list becomes zero.

Having defined the general splitting strategy and code structure, the loop contents will be considered in details. The first step inside the loop body is to extract dimensions through the bounding box for farther filtering and splitting panels. Since panel orientation is added to the definition, rotated panels have two types of dimensions. First type matches with panel sides. These dimensions are used to build middle line and split afterwards. Meanwhile the second type describes area occupied by rotated panel parallel to sheet sides. In its turn, second dimensions are utilized for comparison with sheet dimensions and filtering panels for splitting. All dimensions are extracted from corresponding bounding boxes. The parametric scheme (Figure 4.16) is representing the difference. The bigger bounding box Bbox1 is used for filtering exceeded panels. Meanwhile, the smaller bounding box Bbox2 is originated from the panel before rotation and matches the dimensions of the panel. So, Bbox2 is rotated together with the panel assisting in splitting it if the panel exceeds. The reason why Bbox2 is used for splitting panels is that

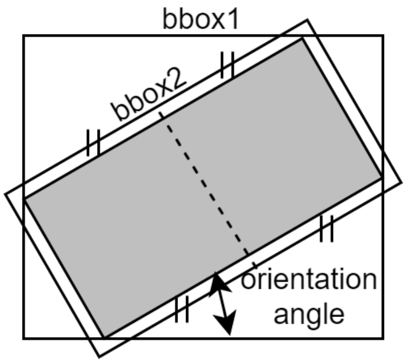


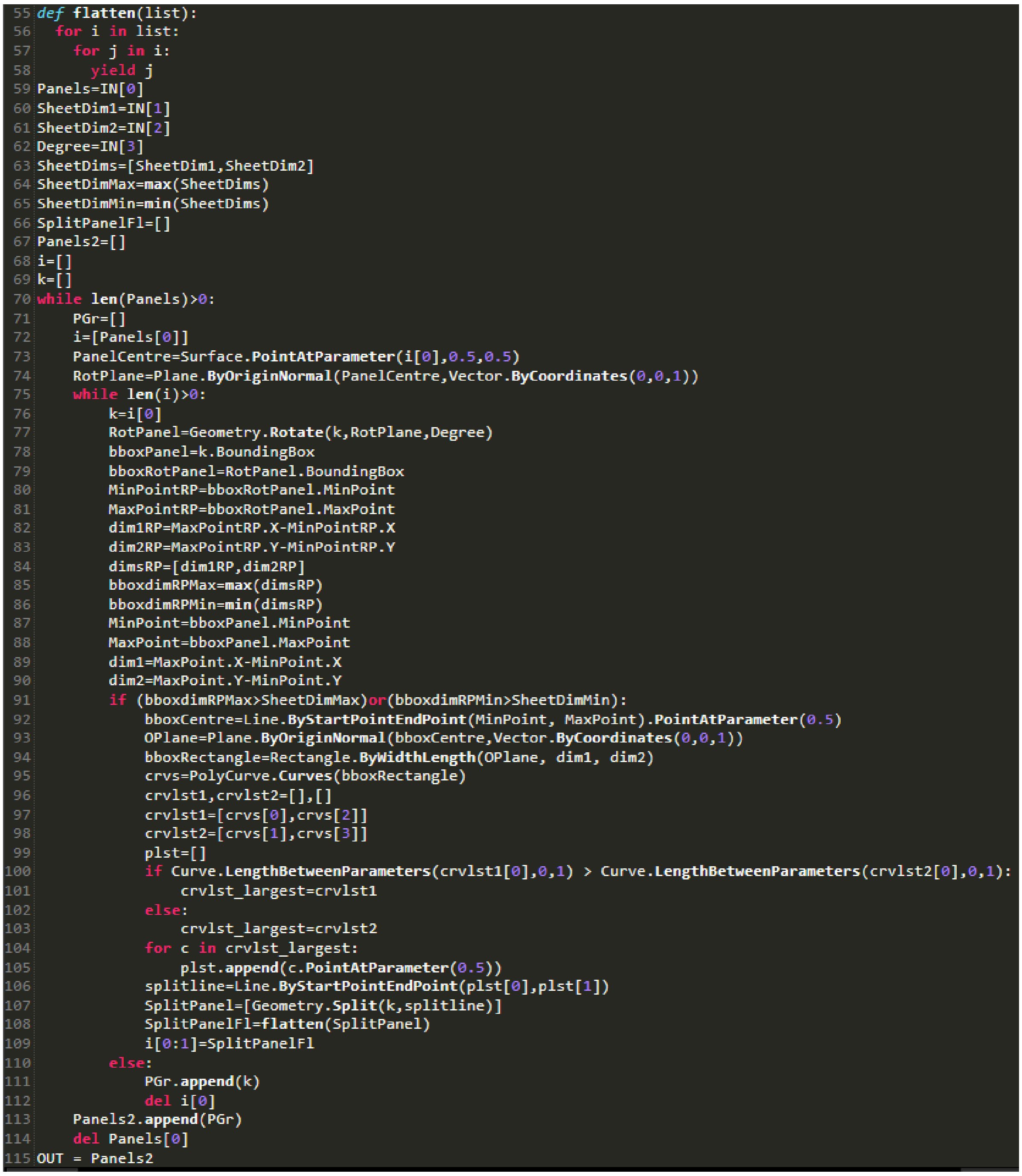
Figure 4.17 - ParametricScheme for splitting panel in loop body (Python)

Bbox1 can be distorted significantly when the panel is rotated. The circumstance of such distortion is wrong side identification for the splitting line as the panel long side may be swiped into a shorter one.

Consequently, the loop body starts with defining Bbox1 (bboxPanel in Figure 4.18, line 78), and Bbox2 (bboxRotPanel in Figure 4.18, line 79) dimensions. So, first of all, rotation plane (RotPlane, Figure 4.18, line 74) is created by method Plane.ByOriginNormal with origin in Panel Centre point defined by another method Surface.PointAtParameter (Figure 4.18, line 73). Then the panel is rotated in rotation plane by Degree input value (Figure 4.18, line 77). Next, bounding boxes are defined with the method i.BoundingBox and then dimensions are determined through the same algorithm as a former splitting graph definition (Figure 4.12) with coordinates subtraction of extremum points. (Figure 4.18, lines 80-90).

Once all the dimensions are determined, the condition “If Else” filters exceeded panels. The distinctive part of the definition is that Bbox1 is used for filtering panels whereas Bbox2 is used for splitting them. (Figure 4.18) The statement for condition “If Else” is comparison between panel Bbox1 dimensions (bboxdimRPMax and bboxdimRPMin) and sheet dimensions (SheetDimMax and SheetDimMin): minimum values and maximum values are compared accordingly. So, if at least one of the Bbox1 dimensions is larger than sheet relative dimension, the panel is split and returns to beginning of loop again. Otherwise, the panel goes to list PGr which groups sub-elements according to input panels. Meanwhile, the panel gets removed from the initial list. (Figure 4.18, lines 91-112)

If the condition is true and the panel exceeds the sheet, the operation code block is executed to split the panel. Before splitting the panel, it is required to define the splitting line which is the middle line of Bbox2 rectangle. Thus, Bbox2 rectangle is created first in the code block. To create the rectangle, method Rectangle.ByWidthLength is used with dimensions of Bbox2 and Origin plane in Bbox2 centre. (Figure 4.18, lines 93-94) Meanwhile, Bbox2 centre is defined by other methods Line.ByStartPointEndPoint and Line.PointAtParameter with MinPoint and MaxPoint of Bbox2. (Figure 4.18, line 92). After the Bbox2 rectangle is created, curves are extracted from rectangle sides and separated into two lists of parallel sides. A further step is to determine which sides to create a middle line between. For this purpose, another condition “If Else” is used comparing property Length of curves in separated lists. So, the panel will be split along the longest side. Consequently, variable “crvlst_largest” accepts the list with the largest value. (Figure 4.18, lines 100-103). Next, internal “For” loop is used to go over curves in the largest list and to write central points of the curves into “plst” list with method list.append. (Figure 4.18, lines 100-101) Finally, the middle line is created as the line between central points from the loop output curves in “plst” list with method Line.ByStartPointEndPoint. Then exceed panel is split with the middle line of Bbox2 rectangle through the method “Geometry.Split”. At the end of code block inside the “while” loop, split panels are inserted into initial “i” list instead of the exceeding panel. Afterwards, resulting list passes again through “while” loop and iteration repeats until all the parts of the input panel will fit and occur in the PGr list. Then all the groups are appended into “Panels2” output list. Once the input “Panels” list is empty, main “while” loop is closed and code finishes with output list Panels2 filled by split panels fitting sheet size. (Figure 4.18, line 115)



```python
def flatten(list):
  for i in list:
    for j in i:
      yield j
Panels=IN[0]
SheetDim1=IN[1]
SheetDim2=IN[2]
Degree=IN[3]
SheetDims=[SheetDim1,SheetDim2]
SheetDimMax=max(SheetDims)
SheetDimMin=min(SheetDims)
SplitPanelFl=[]
Panels2=[]
i=[]
k=[]
while len(Panels)>0:
    PGr=[]
    i=[Panels[0]]
    PanelCentre=Surface.PointAtParameter(i[0],0.5,0.5)
    RotPlane=Plane.ByOriginNormal(PanelCentre,Vector.ByCoordinates(0,0,1))
    while len(i)>0:
        k=i[0]
        RotPanel=Geometry.Rotate(k,RotPlane,Degree)
        bboxPanel=k.BoundingBox
        bboxRotPanel=RotPanel.BoundingBox
        MinPointRP=bboxRotPanel.MinPoint
        MaxPointRP=bboxRotPanel.MaxPoint
        dim1RP=MaxPointRP.X-MinPointRP.X
        dim2RP=MaxPointRP.Y-MinPointRP.Y
        dimsRP=[dim1RP,dim2RP]
        bboxdimRPMax=max(dimsRP)
        bboxdimRPMin=min(dimsRP)
        MinPoint=bboxPanel.MinPoint
        MaxPoint=bboxPanel.MaxPoint
        dim1=MaxPoint.X-MinPoint.X
        dim2=MaxPoint.Y-MinPoint.Y
        if (bboxdimRPMax>SheetDimMax)or(bboxdimRPMin>SheetDimMin):
            bboxCentre=Line.ByStartPointEndPoint(MinPoint, MaxPoint).PointAtParameter(0.5)
            OPlane=Plane.ByOriginNormal(bboxCentre,Vector.ByCoordinates(0,0,1))
            bboxRectangle=Rectangle.ByWidthLength(OPlane, dim1, dim2)
            crvs=PolyCurve.Curves(bboxRectangle)
            crvlst1,crvlst2=[],[]
            crvlst1=[crvs[0],crvs[2]]
            crvlst2=[crvs[1],crvs[3]]
            plst=[]
            if Curve.LengthBetweenParameters(crvlst1[0],0,1) > Curve.LengthBetweenParameters(crvlst2[0],0,1):
                crvlst_largest=crvlst1
            else:
                crvlst_largest=crvlst2
            for c in crvlst_largest:
                plst.append(c.PointAtParameter(0.5))
            splitline=Line.ByStartPointEndPoint(plst[0],plst[1])
            SplitPanel=[Geometry.Split(k,splitline)]
            SplitPanelFl=flatten(SplitPanel)
            i[0:1]=SplitPanelFl
        else:
            PGr.append(k)
            del i[0]
    Panels2.append(PGr)
    del Panels[0]
OUT = Panels2
```

Figure 4.18 - Body of Python script defining splitting loop over input panels

Once all the Revit component's geometry is panelized, the guide component can be reconstructed in Revit space with defined panels for manufacturing convenience and assembling. The logic of this script part is straightforward. Split panels are transformed from XY plane back to origin planes and are baked with Direct Shape. (Figure 4.19) To transform geometry, coordinate systems are used which are defined by origin points and planes of elements. The important note here is that quantity of split panels is different from the initial number of panels. Therefore, split panels were grouped in Python code prior to reconstruction. Within these groups, split panels are transformed by the same coordinate system. After panels are transformed, they are modified into solids with Surface.Thicken node. Next, Revit instances are baked with DirectShape.ByGeometry node. The category is Generic Models. Finally, every instance panel is filled with Mark parameter value that matches with the tag of the panel in layout. (Figure 4.20) Adding a tag to panels in the layout will be considered in fourth milestone.

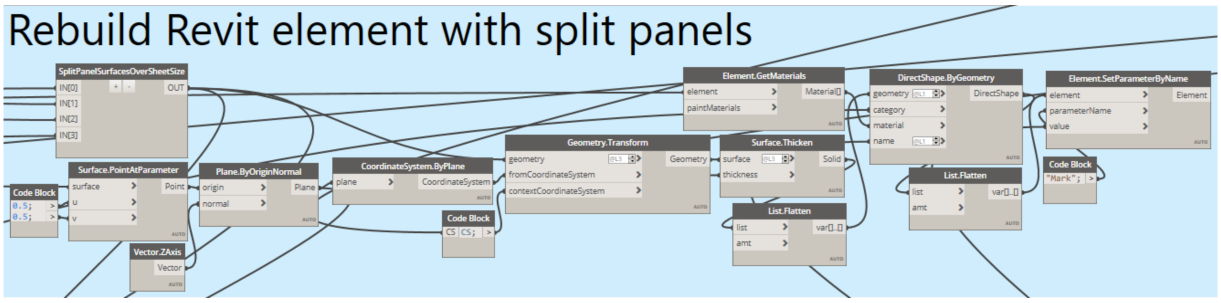


Figure 4.19 - Rebuilding Revit component as guide for panels assembling. Graph definition

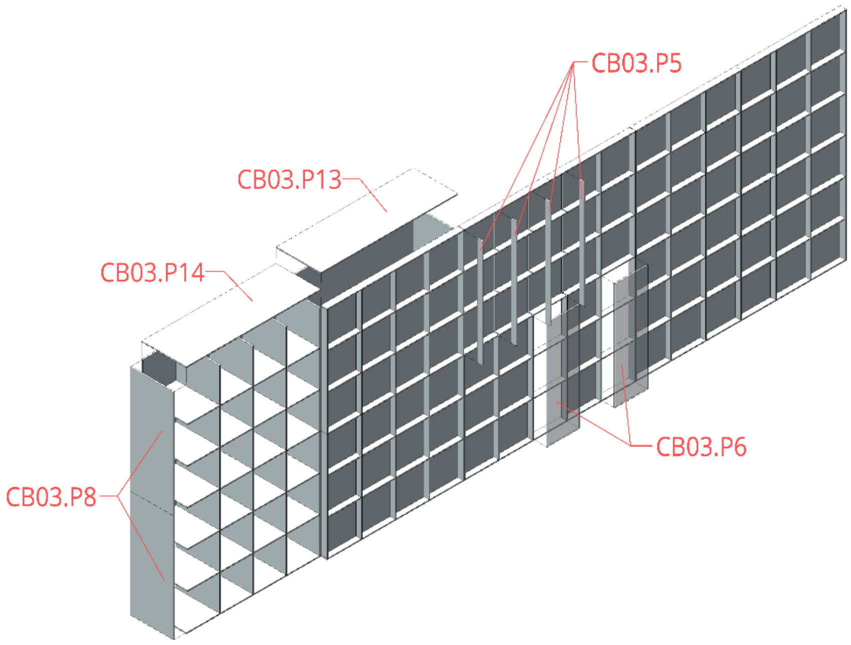


Figure 4.20 - Guide Revit component for panels assembling. Revit View port.

4.2.4. 3rd Milestone: Packing in Sheets

In the third part of the script definition, panels are packed in sheets. The workflow is divided into several local tasks. First of them is sheets creation. Then panel bounding boxes are packed into sheets. Next, panels placed in sheet space according to their bounding boxes packing. Finally, panels are verified if their bounding boxes are rotated after packing. If so, such panels are rotated consequentially. As output, all the panels should be packed on XY plane within sheets in Dynamo.

To begin with, sheets are created from two input dimension values. The number of sheets is defined through the division of Sum area of all the panels by sheet area. The Sum area of panels is multiplied by coefficient 1,6 which compensates all unoccupied space probably emerged between packed elements on sheets. Then Math.Ceiling node is used to round the number value to the first greater integer. Once the number of sheets is defined, sheet rectangle is created and can be placed with an array of distance values to form the set of sheets. The number of sheets determines the number of values in the distance array. Meanwhile, the input width multiplied by 1.1 determines step in the array. Finally, sheet geometry is translated and form the set of sheets for further layout. (Figure 4.21)

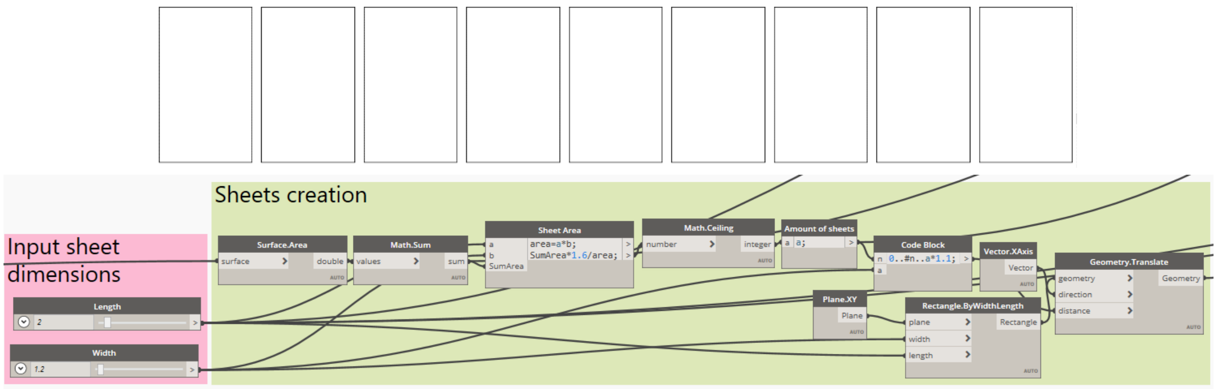


Figure 4.21 - Sheets creation for packing panels in Dynamo. Visualization on top.

Next part of script packs panels on sheets. RefineryToolkits package is used for packing function as it appears (from the previous case study experience) to better suit for the packing of elements in Revit. Unlike in the second case study, when views were laid out on sheets and could not be rotated, panels should be packed without rotation constrain. Therefore, RefineryToolkits node Packing.PackRectagles fits the best for the current task. To perform the task, panel bounding boxes are packed first and then panels are moved inside the bounding boxes. This transition is performed because of Packing.PackRectangles node excepts rectangles as its input for packing. Thus, rectangles are created from bounding box dimensions with additional control of input "gap" parameter which determines gap distance between panels on sheet. For Container input of Packing.PackRectangles node rectangles created from sheets are used. (Figure 4.22) There are three different strategies used for packing in RefineryToolkits. One of them is Rectangle Area Strategy which picks the free area that is the smallest in the area to place the next rectangle into. (Figure 4.23, on top) Another one is Rectangle Long Side Strategy which packs the next rectangle into the free area where the length of the shorter leftover side is minimized. (Figure 4.23, in the middle) The last one is the opposite strategy to the second, Rectangle

Short Side Strategy, and packs next rectangle into the free area where the length of the longer leftover side is minimized. (Figure 4.23, on bottom) After exploring layout results the Rectangle Short Side Strategy is selected because it utilizes fewer sheets and more space within the sheet. As a result, the Rectangle Short Side Strategy allowed making the most compact layout.

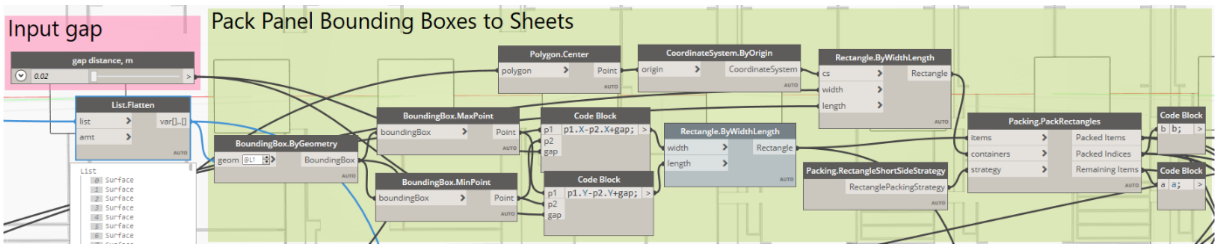


Figure 4.22 - Packing panel bounding boxes into sheets. Definition

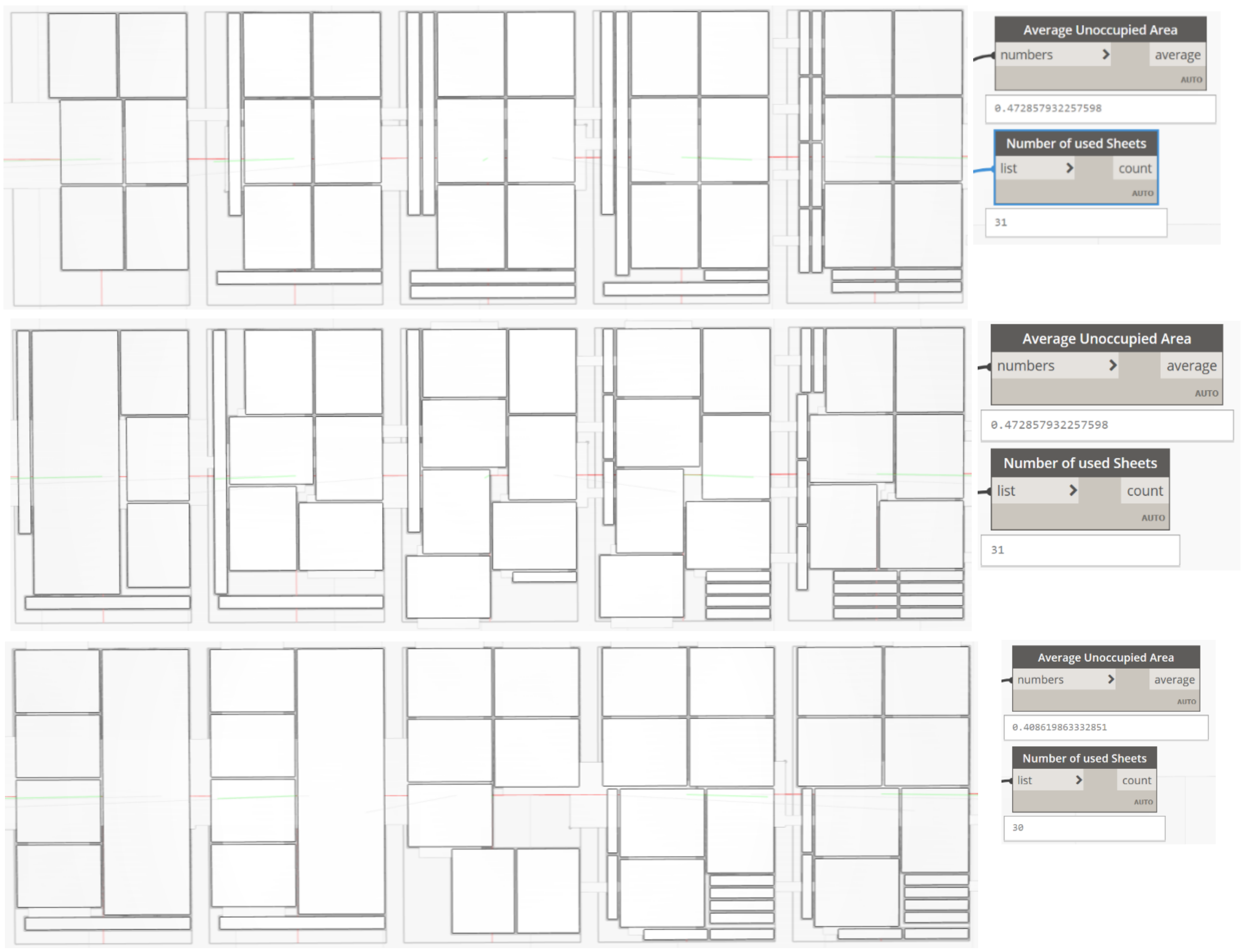


Figure 4.23 – Panelization optimized layout with Area Packing Strategy (top -RefineryToolkits), LongSide Packing Strategy (middle -RefineryToolkits), ShortSide Packing Strategy (bottom - RefineryToolkits)

Once bounding boxes are laid out on sheets, panels can be placed accordingly. To place panels, vectors are created between central points of panels and central points of their packed bounding boxes.

Then panels are translated to the packing position. It is notable, that after the translation of panels, some of them are not rotated, unlike their bounding boxes. This happens because there is no orientation relationship established so far between panels and bounding boxes. (Figure 4.24, orientation errors are marked in red frames) To define this relationship, the width parameter is compared between initial bounding boxes and their packed versions. Then the filter is applied for panels to determine those of them for rotation. Finally, panels are rotated and remerged into the single list again with the rest. (Figure 4.25)

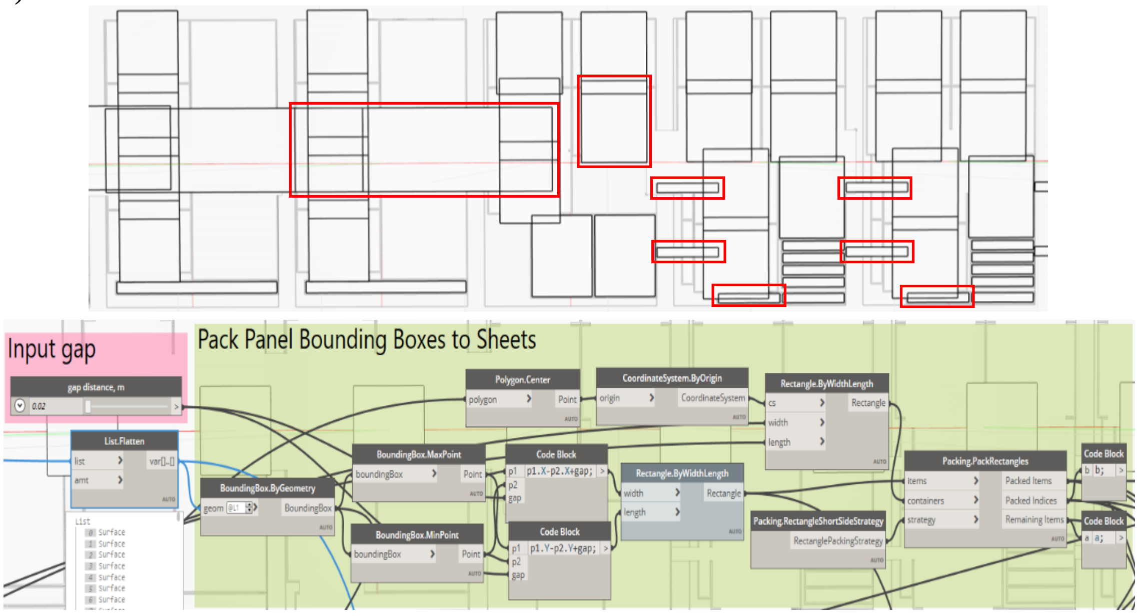


Figure 4.24 - Placing panels to sheets according to their bounding box layout (before rotation) Visualization on top. Orientation error in red frames

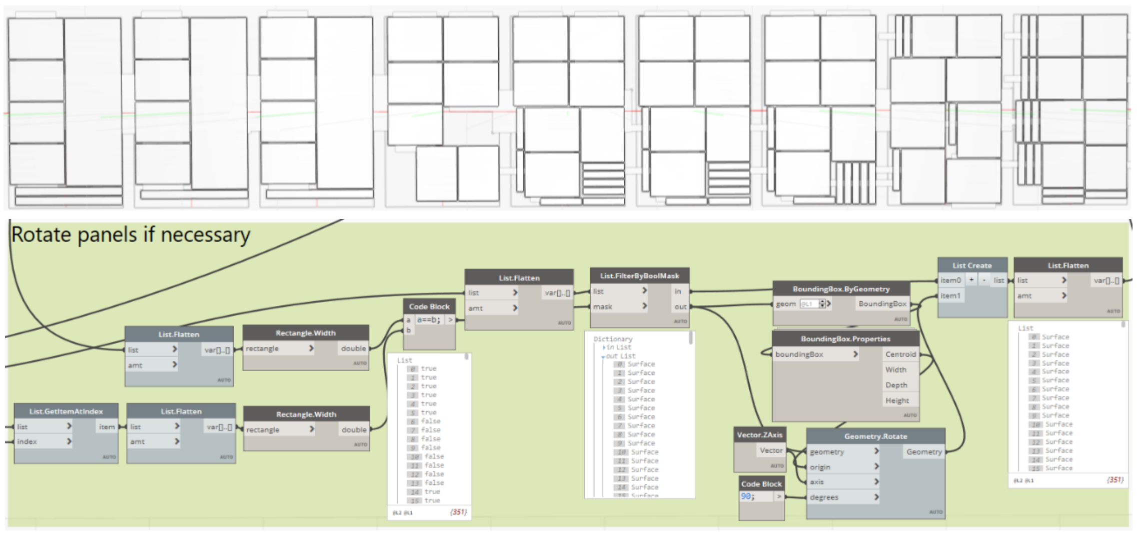


Figure 4.25 - Rotation of panels definition (Dynamo). Result on top.

4.2.5. 4th Milestone: Drawing View Generation

The last part of the script definition is drawing view generation with marked panels and legend. First of all, drafting view is created with the node DraftingView.ByName. The name of view inherits the Mark parameter value of Revit laid out component.

Then contours of panels are extracted with the node Surface.PerimeterCurves and are baked on the drafting view. (Figure 4.26, Baking contours to drafting view) Finally, the view scale is defined by input value. It helps to control the size of the panel tags on view. (Figure 4.26, Set View Scale)

Once the drafting view is created and panels contours are baked, it is time to tag them for further manufacturing. To tag panels, a dictionary is supposed to be created which will associate the size of the

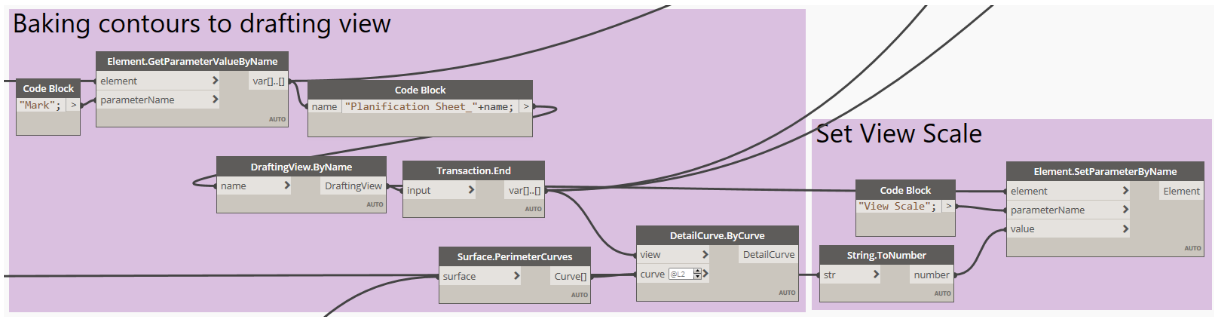


Figure 4.26 - Creation drafting view with baked panel contours laid out on sheets

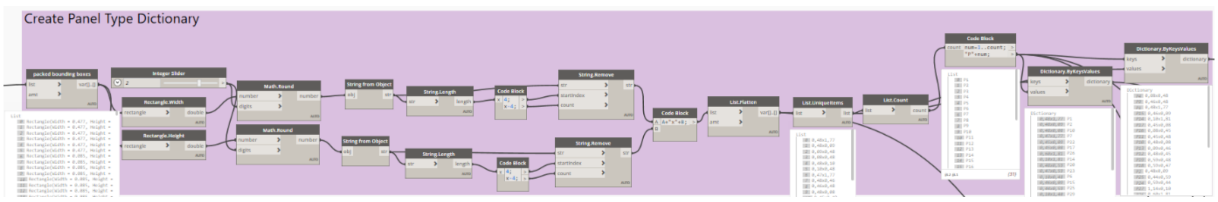
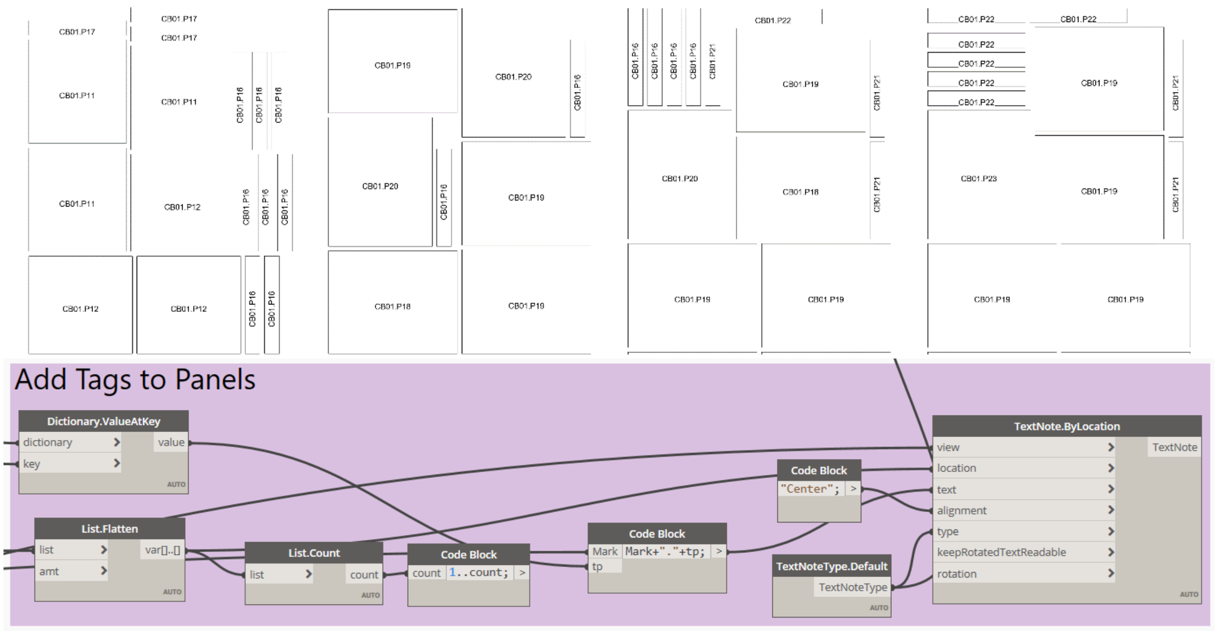


Figure 4.27 - Creating of Panel Type Dictionary (Dynamo)

panel with the name of the type. Consequentially, sizes are converted from dimensions to strings, joined and unique values are determined. Then names of types are defined through the sequence of numbers from the count of unique values with the prefix “P”. Then two transposed dictionaries are created with node Dictionary.ByKeysValues. One with size keys and type names values is created to tag panels. Meanwhile, another one with type names keys and size values is for the Legend. (Figure 4.27)

In the next part, tags are added to all the panels in the drafting view with text inherited from the component instance parameter “Mark” and type name extracted from the dictionary. Tags are added with node TextNote.ByLocation. For the location, polygon centres of packed bounding boxes are used to align tags. The result is seen in Figure 4.28.



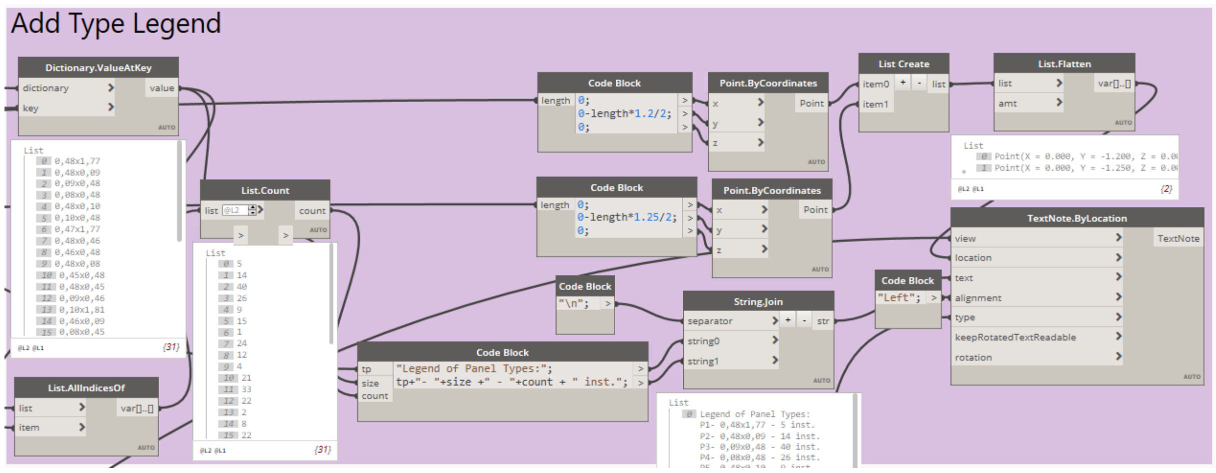
```
"Center";
```

```
1..count;
```

```
Mark+"."+tp;
```

Figure 4.28 - Adding Tags to Panels from Dictionary (Dynamo). Tags in Revit (top)

In the end, Type Legend is created with the same node TextNote.ByLocation. For the location, Origin point with coordinates [0,0,0] with shift down equal to input length parameter of sheet multiplied by 1,25 is used to place Legend under the sheet row in the beginning. Text is joined string from the type name, panel size from the created dictionary, and the instances count value. The definition is seen in Figure 4.29.



```
List
  0 0,48x1,77
  1 0,48x0,09
  2 0,09x0,48
  3 0,08x0,48
  4 0,48x0,10
  5 0,10x0,48
  6 0,47x1,77
  7 0,48x0,46
  8 0,46x0,48
  9 0,48x0,08
  10 0,45x0,48
  11 0,48x0,45
  12 0,09x0,46
  13 0,10x1,81
  14 0,46x0,09
  15 0,08x0,45
@L2 @L1                {31}
```

```
List
  0 5
  1 14
  2 40
  3 26
  4 9
  5 15
  6 1
  7 24
  8 12
  9 4
  10 21
  11 33
  12 22
  13 2
  14 8
  15 22
@L2 @L1                {31}
```

```
0;
0-length*1.2/2;
0;
```

```
0;
0-length*1.25/2;
0;
```

```
List
  0 Point(X = 0.000, Y = -1.200, Z = 0.0
  1 Point(X = 0.000, Y = -1.250, Z = 0.0
@L2 @L1                {2}
```

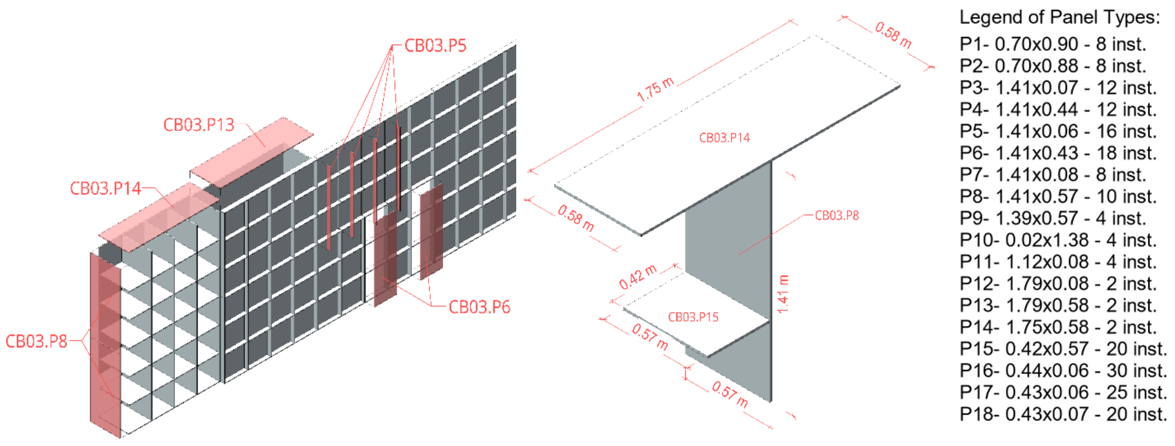
```
"\n";
```

```
"Legend of Panel Types:";
tp+"- "+size +" - "+count + " inst.";
```

```
"Left";
```

```
List
  0 Legend of Panel Types:
    P1- 0,48x1,77 - 5 inst.
    P2- 0,48x0,09 - 14 inst.
    P3- 0,09x0,48 - 40 inst.
    P4- 0,08x0,48 - 26 inst.
```

Figure 4.29 - Adding Type Legend in Dynamo



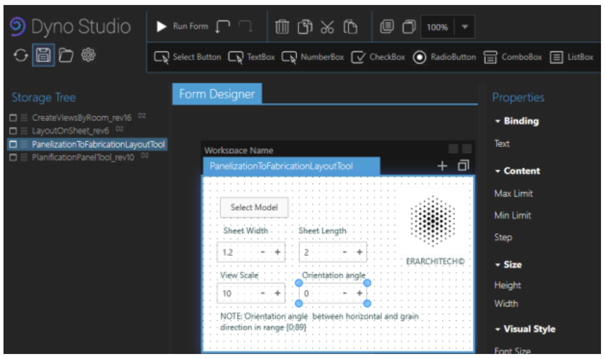
Legend of Panel Types:
P1- 0.70x0.90 - 8 inst.
P2- 0.70x0.88 - 8 inst.
P3- 1.41x0.07 - 12 inst.
P4- 1.41x0.44 - 12 inst.
P5- 1.41x0.06 - 16 inst.
P6- 1.41x0.43 - 18 inst.
P7- 1.41x0.08 - 8 inst.
P8- 1.41x0.57 - 10 inst.
P9- 1.39x0.57 - 4 inst.
P10- 0.02x1.38 - 4 inst.
P11- 1.12x0.08 - 4 inst.
P12- 1.79x0.08 - 2 inst.
P13- 1.79x0.58 - 2 inst.
P14- 1.75x0.58 - 2 inst.
P15- 0.42x0.57 - 20 inst.
P16- 0.44x0.06 - 30 inst.
P17- 0.43x0.06 - 25 inst.
P18- 0.43x0.07 - 20 inst.

Figure 4.30 – Varification of Panel Types with 3d Guide model

4.2.6. Panelization to Fabrication.

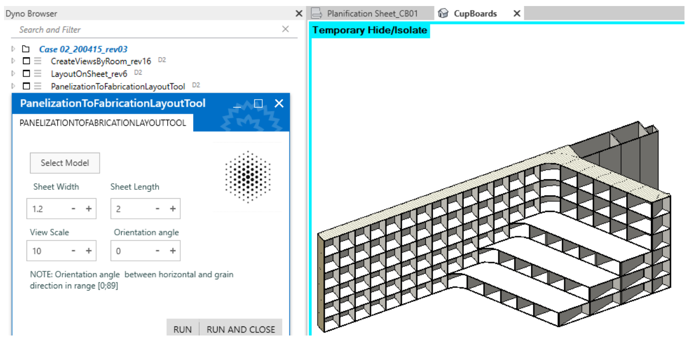
USER INTERFACE.

The written script definition needs a user interface for interaction with the end-user designer. The user interface is written with Dyno Studio platform which was implemented in the previous case studies as well. So, Form was created with Form Designer with six buttons embed: five of them binded with input parameters such as Sheet Width, Sheet Length, View Scale, Gap Value and Orientation Angle; the sixth button is Component Selector for Panelization. (Figure 4.31) To run the script through UI, Dyno plugin shell be installed. The UI form is synchronized with Revit through Dyno Browser and can be called by double-clicking as well as its dialogue window accessed when running script in Dynamo editor. (Figure 4.32)



NOTE: Orientation angle between horizontal and grain direction in range [0;89]

Figure 4.31 - UI Form design in Dyno Studio



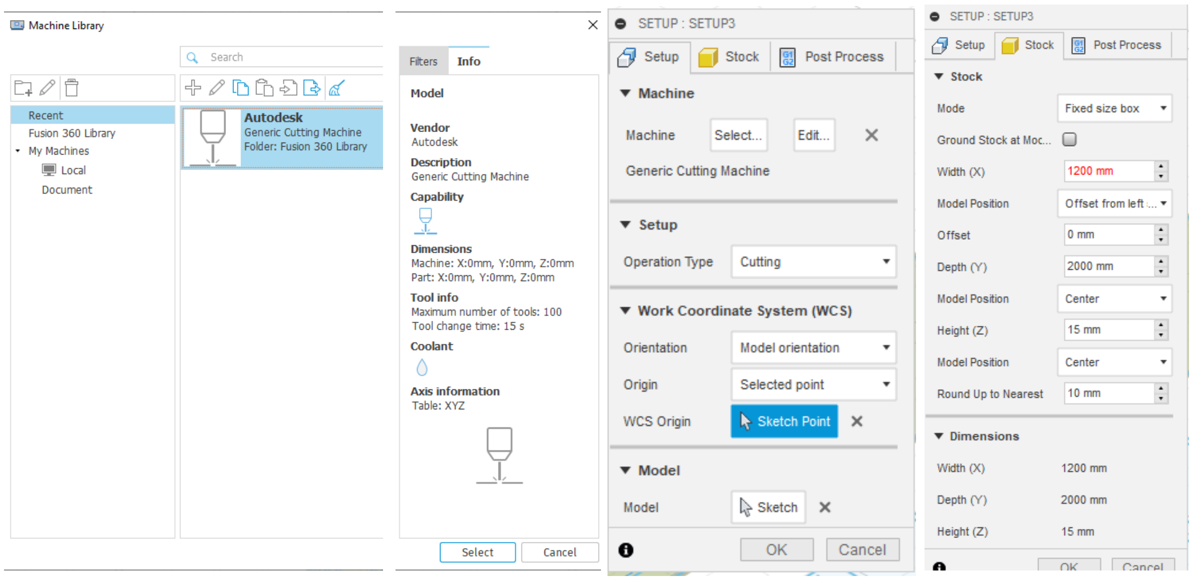
NOTE: Orientation angle between horizontal and grain direction in range [0;89]

Figure 4.32 - UI Form called in Revit

EXPORT TO G-CODE.

Once the script is written and the user interface is set up, we can export panels layout to G-Code for farther manufacturing. To generate G-Code, CAM software is needed where toolpaths can be specified and the cutting tool can be chosen from the list to set up proprietary M-Code instructions for a specific machine. Fusion 360 was chosen for exporting G-Code as it has quite an intuitive CAM tool and extensive settings.

Consequently, the drafting view with the layout was exported in .dwg format to Fusion 360. Next, in Manufacture mode, setup first is defined with machine selected, operation type (cutting), WCS (Work Coordinate System), and layout contours for cutting. (Figure 4.33, machine selection on the left; operation type, WCS, and contours selection in the middle) Next, stock properties are determined such as sheet dimensions and position relevant to the layout. (Figure 4.33, on the right) Once setup is defined, toolpaths can be simulated by hitting the “Simulate” button in the Actions tab menu. In simulation mode, it is possible to verify the cutting process and playback it if needed. After verification of the cutting process, the G-Code file is exported with a specific machine and vendor preset configuration selected from the drop-down menu. As a result, a G-Code file in format .nc is generated. (Figure 4.34, right) The exporting process as simple as described above. Though, the workflow still allows only a single sheet per operation. Therefore, manual iteration procedure is needed to export each sheet separately.


Machine Library

Recent
Fusion 360 Library
My Machines
Local
Document

Autodesk
Generic Cutting Machine
Folder: Fusion 360 Library

Model

Vendor
Autodesk
Description
Generic Cutting Machine
Capability
Dimensions
Machine: X:0mm, Y:0mm, Z:0mm
Part: X:0mm, Y:0mm, Z:0mm
Tool info
Maximum number of tools: 100
Tool change time: 15 s
Coolant
Axis information
Table: XYZ

SETUP : SETUP3

Machine
Machine: Select... Edit...
Generic Cutting Machine

Setup
Operation Type: Cutting

Work Coordinate System (WCS)
Orientation: Model orientation
Origin: Selected point
WCS Origin: Sketch Point

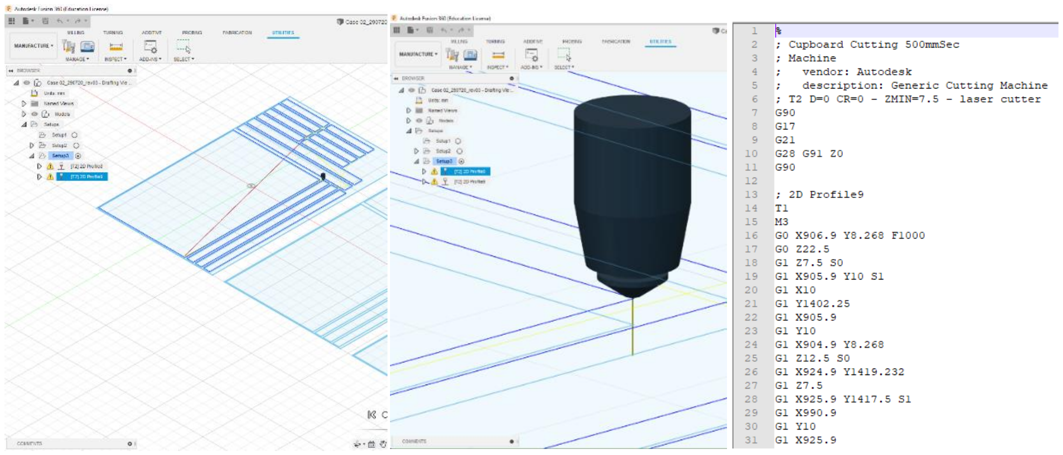
Model
Model: Sketch

SETUP : SETUP3

Stock
Mode: Fixed size box
Ground Stock at Moc...
Width (X): 1200 mm
Model Position: Offset from left ...
Offset: 0 mm
Depth (Y): 2000 mm
Model Position: Center
Height (Z): 15 mm
Model Position: Center
Round Up to Nearest: 10 mm

Dimensions
Width (X): 1200 mm
Depth (Y): 2000 mm
Height (Z): 15 mm

Figure 4.33 – Manufacture setup in Fusion 360. Machine selection (left), setup bar(middle), stock bar (right)



```
%
; Cupboard Cutting 500mmSec
; Machine
;   vendor: Autodesk
;   description: Generic Cutting Machine
; T2 D=0 CR=0 - ZMIN=7.5 - laser cutter
G90
G17
G21
G28 G91 Z0
G90

; 2D Profile9
T1
M3
G0 X906.9 Y8.268 F1000
G0 Z22.5
G1 Z7.5 S0
G1 X905.9 Y10 S1
G1 X10
G1 Y1402.25
G1 X905.9
G1 Y10
G1 X904.9 Y8.268
G1 Z12.5 S0
G1 X924.9 Y1419.232
G1 Z7.5
G1 X925.9 Y1417.5 S1
G1 X990.9
G1 Y10
G1 X925.9
```

Figure 4.34 - Simulation of toolpaths in Fusion 360 (left), tool pathing contours (middle), G-Code (right)

4.3. CONCLUSION

The primary conclusion from the case study investigation is that completely digital workflow BIM-to-fabrication is possible. There are many implementation cases which are evidence of this statement. On every design stage, there is a digital tool allowing to extend information and details based on the previous stage model retaining a connection between them. Consequently, technologies provide a full chain for BIM-to-fabrication transition.

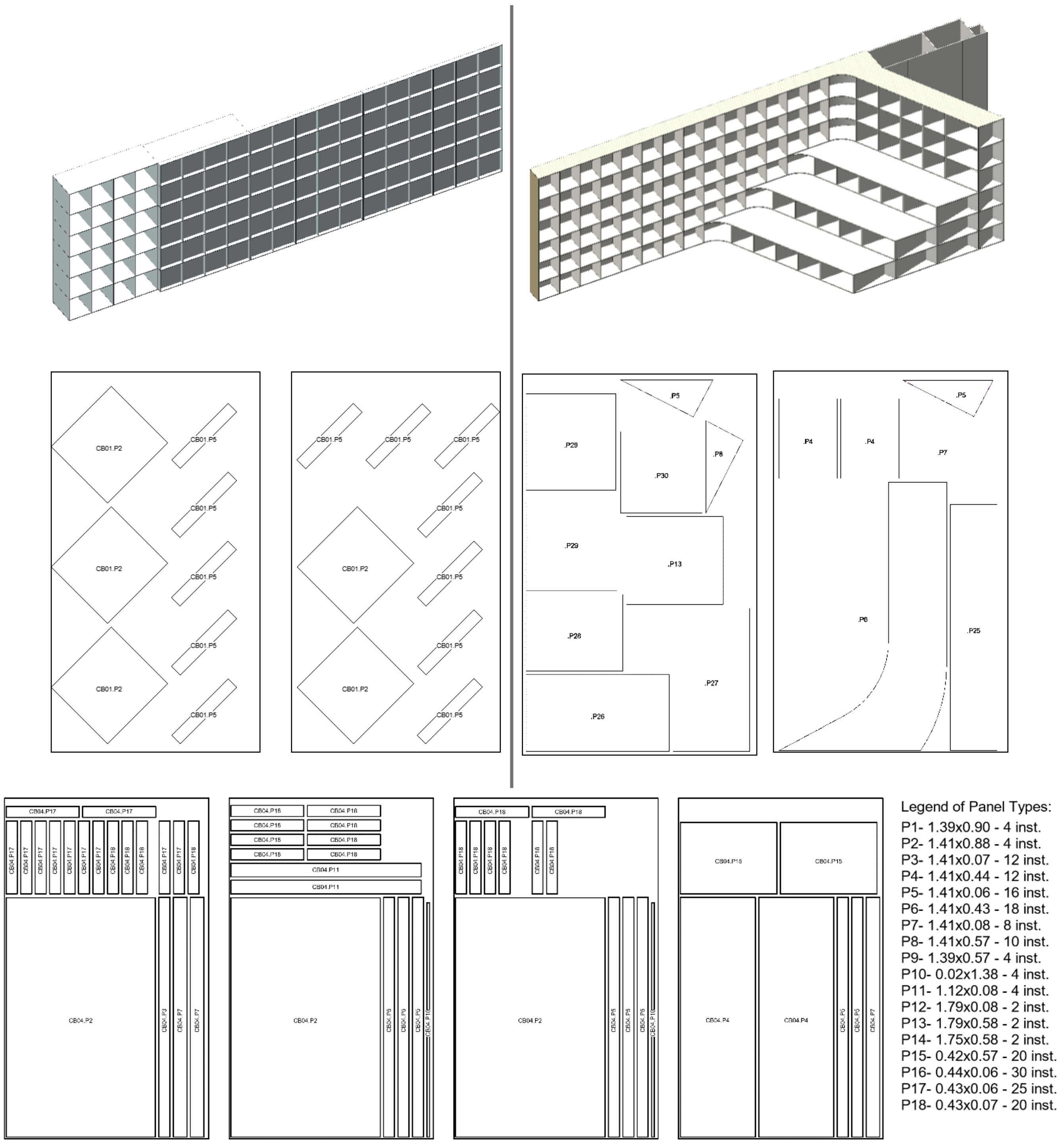
Though there are transition issues called by the accuracy and reliability of the BIM model due to the lack of focus on fabrication during the design stages. Therefore, the design needs to take fabrication and assembly seriously at an early stage of the process, to avoid unexpected delays and costs later on. Another issue is that all CNC machines have proprietory M-Code part which requires the use of the third-party tools for export G-Code instruction specific to the machine. Exceptionally, some vendors provide their tools integrated into the BIM authoring platforms. One more issue is the need for programming skills to create accurate model setup and layout for fabrication.

As a solution, the concept of DfMA is used for integration of material-informed parametric model from the early design stages. Such an approach is used to predict and tune the final shape and translate a design geometry to the material information required for production. There were presented two cases at different scales of design - pavilion and airport terminal. Both cases demonstrate the successful implementation of this approach from scratch.

However, it still often needs to prepare the already existing non-parametric design for fabrication. In such a case, scripting is used to adapt the model. Consequently, such a case is investigated in this chapter - panelization of existing BIM component for digital fabrication and preparing this for laser cutting.

As a result, Panelization tool is elaborated in Dynamo editor written in DesignScript both by associative visual programming means as well as by imperative Python scripting means. Intentionally, the tool selects the component from Revit model. Next, based on specified inputs such as sheet dimensions, the orientation angle of panels, and the gap between neighbour panels, the script performs panelization of component geometry and layout panels on sheet. Amount of sheets and panel layout are optimized with RefineryToolkits. In the end, drafting view is generated with panel layout ready for export to CAM software for farther creation of toolpaths and G-Code generation for CNC machine instructions. Notably, the tool has some limitations such as mutual cut pattern dependency between parts as it was elaborated under the specific design needs of the company. Though it works well if such a pattern is determined before script development.

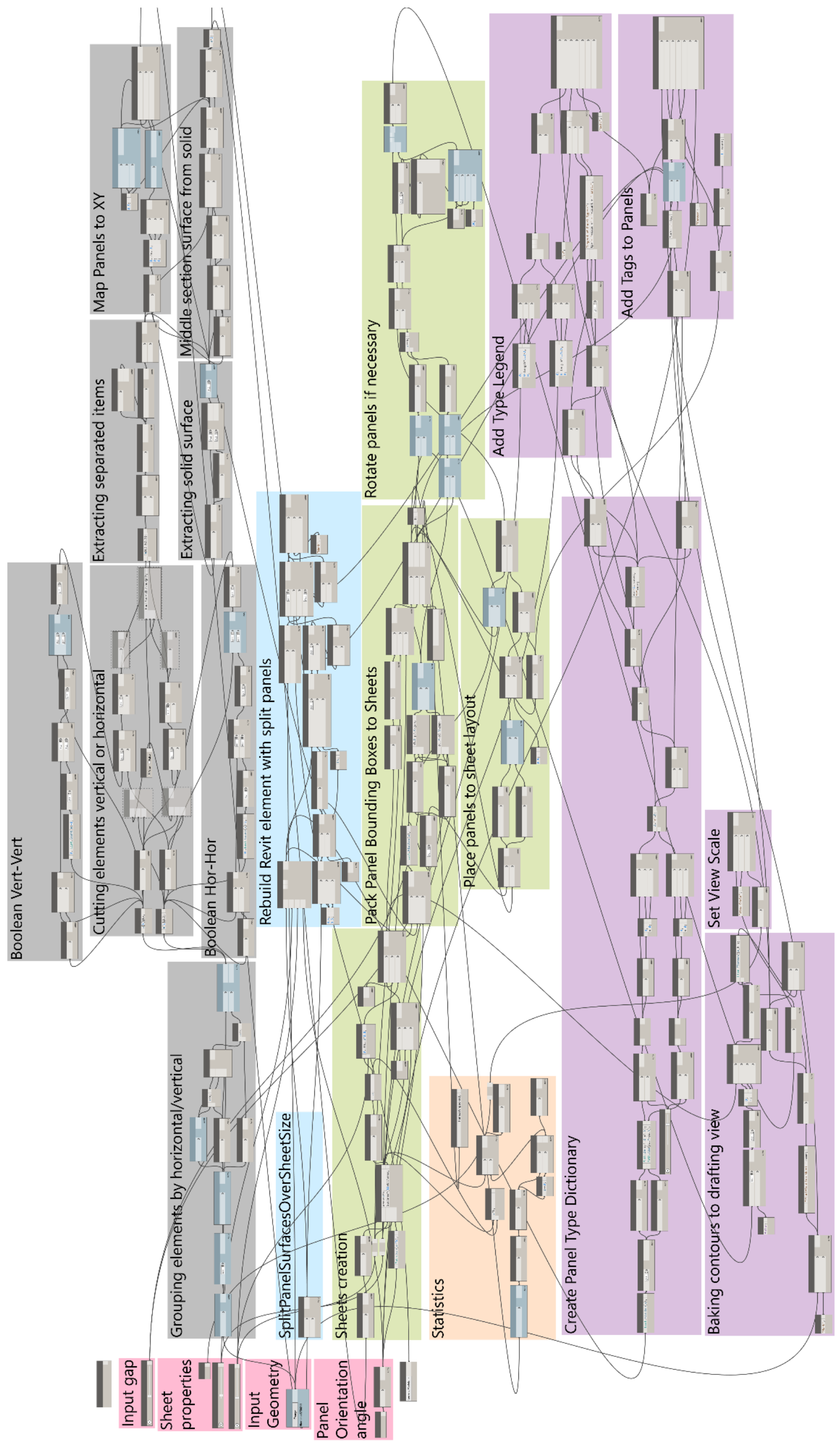
In nutshell, prefabrication is introduced but still an ‘alien’ concept in many parts of the AEC industry, as it needs deep investigation and parametric integration from the early design stages. BIM is rapidly developing but the continuous use and coordination of digital data models need to be extended all the way from design to digital fabrication and assembly processes. The modelling paradigms of BIM are largely concerned with building components but do not define fabrication inputs for the design. Multi-scalar parametric modelling is not yet implemented, either in the workflows, tools or standards. The future development should be focused on DfMA approach and parametric implementation.



Legend of Panel Types:

P1- 1.39x0.90 - 4 inst.
P2- 1.41x0.88 - 4 inst.
P3- 1.41x0.07 - 12 inst.
P4- 1.41x0.44 - 12 inst.
P5- 1.41x0.06 - 16 inst.
P6- 1.41x0.43 - 18 inst.
P7- 1.41x0.08 - 8 inst.
P8- 1.41x0.57 - 10 inst.
P9- 1.39x0.57 - 4 inst.
P10- 0.02x1.38 - 4 inst.
P11- 1.12x0.08 - 4 inst.
P12- 1.79x0.08 - 2 inst.
P13- 1.79x0.58 - 2 inst.
P14- 1.75x0.58 - 2 inst.
P15- 0.42x0.57 - 20 inst.
P16- 0.44x0.06 - 30 inst.
P17- 0.43x0.06 - 25 inst.
P18- 0.43x0.07 - 20 inst.

Figure 4.35 – Panelization Results in Revit. Grain Orientation method in the middle left, Best Fit method in bottom row. Revit 1st Type Cupboard component on top left, 2nd Type on top right. Legend on bottom right. *The resulting numbers of types are different between methods.


Map Panels to XY
Middle section surface from solid
Extracting separated items
Extracting solid surface
Boolean Vert-Vert
Cutting elements vertical or horizontal
Boolean Hor-Hor
Grouping elements by horizontal/vertical
Rebuild Revit element with split panels
SplitPanelSurfacesOverSheetSize
Rotate panels if necessary
Pack Panel Bounding Boxes to Sheets
Sheets creation
Place panels to sheet layout
Statistics
Add Type Legend
Add Tags to Panels
Create Panel Type Dictionary
Set View Scale
Baking contours to drafting view
Input gap
Sheet properties
Input Geometry
Panel Orientation angle

1st Milestone (Grey) | 2nd Milestone (Blue) | 3rd Milestone (Green) | 4th Milestone (Violet)

Figure 4.36 - Full Graph definition

This page is intentionally left blank

5. CASE 04. TEST AIM MODEL

5.1. INTRODUCTION

In this chapter, the main objective is to testify the proposed methodology based on prior theory investigation and exploited former cases studies within unified workflow. This chapter is intended to conclude the practical part of the dissertation with consolidating comprehensive vision of the subject. Through the experience gathered from the applied AIM methodology in full range workflow, it is also expected to leverage competency and enhance practical knowledge of the subject. Primarily two options were considered for the testing investigated methodology. Firstly, there was a suggestion to test different aspects based on existing projects in the partner company picking particular problems for implementing AIM tools as solutions separately. Another option was to explore a single design project throughout different stages, examining AIM utilized as leading methodology controlling all the main aspects of the project. The first one advantages from focusing on particular problem shooting, hence the investigation should be more detailed. From the other hand, the second option gives unique opportunity to reveal and examine the potential of AIM in the full range establishing design workflow instead of isolated integrating into the existing one. Though both options are acceptable and together provide an exhaustive subject overview. As a result, first, the former option was chosen for case studies investigated within company practical projects. Then, the latter approach was applied as underscore emphasis summarizing former endeavours and examining complete AIM workflow in practice.

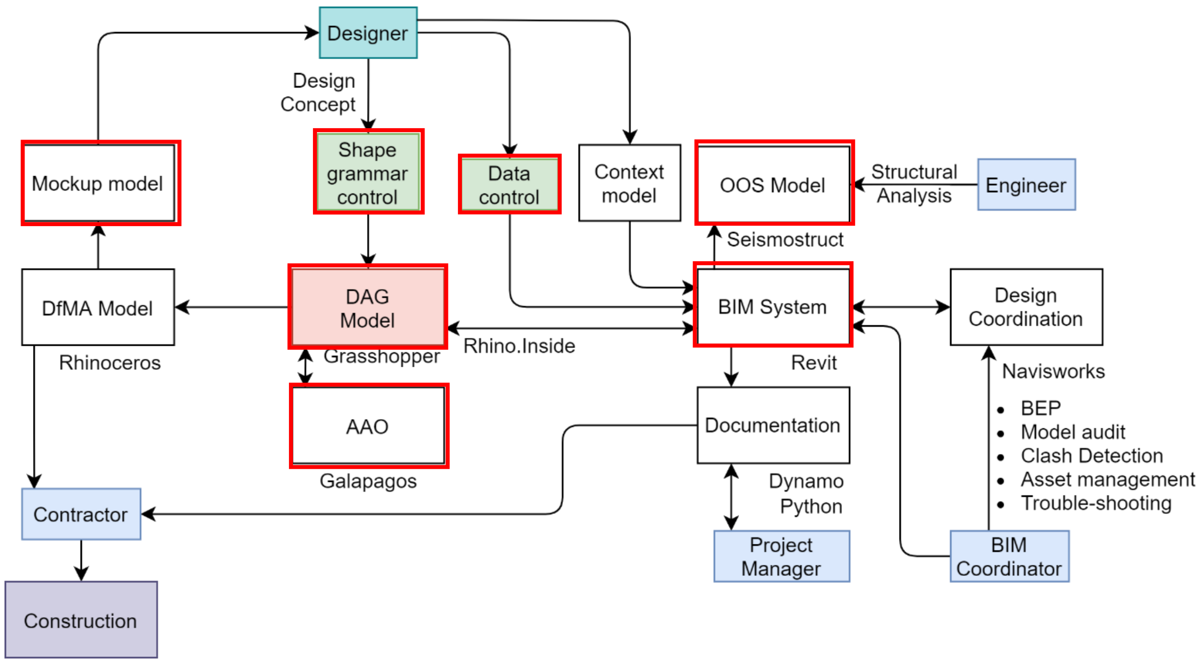


Figure 5.1 – Test AIM workflow: embraced modules in this study are in red frames

Subsequently, the test model is developed from the scratch covering main aspects of AIM. The embraced range of aspects includes the creation of Graph Model (DAG) as Central Source System; its optimization; generation of BIM model for further project development, simulations and coordination; and DfMA Model for mockups and fabrication. Main Graph is developed and optimized within Grasshopper GAE as the most matured GAE offering a diversity of addons for controlling the computational system. DfMA model is controlled from the same graph and connected with CAM software through Rhinoceros. Finally, the BIM model is produced in Revit been directly controlled from the former graph as well with Rhino.Inside plugin. Further based on created models, various testing tasks are performed, such as export to engineering software for seismic analysis Seismostruct and coordination software. The workflow is presented in Figure 5.1.

5.2. CASE STUDY. AIM MODEL

Phoenix International Media Center, built by local company BIAD Ufo in Beijing, 2012, was selected as prototype project for this case study. (Figure 5.2) Phoenix International Media Center locates in the southwest corner of Chaoyang Park; the site area is 1.8 hectares. The total floor area of the building is 65,000m2 with a height of 55m. Apart from the media office, the broadcasting studios and the production offices, the building provides abundant of open spaces for the public to get interactive experiences, which expresses the unique operation concept of Phoenix Media. The logic of the design concept is to create an ecological environment shell embraces the Individual functional spaces as a building-in-building concept. The two independent office towers under the shell generate many shared public spaces. In the east and west parts of the shared spaces, there are continuous steps, landscape platforms, sky ramps and crossing escalators which fill the building of energetic and dynamic spaces.

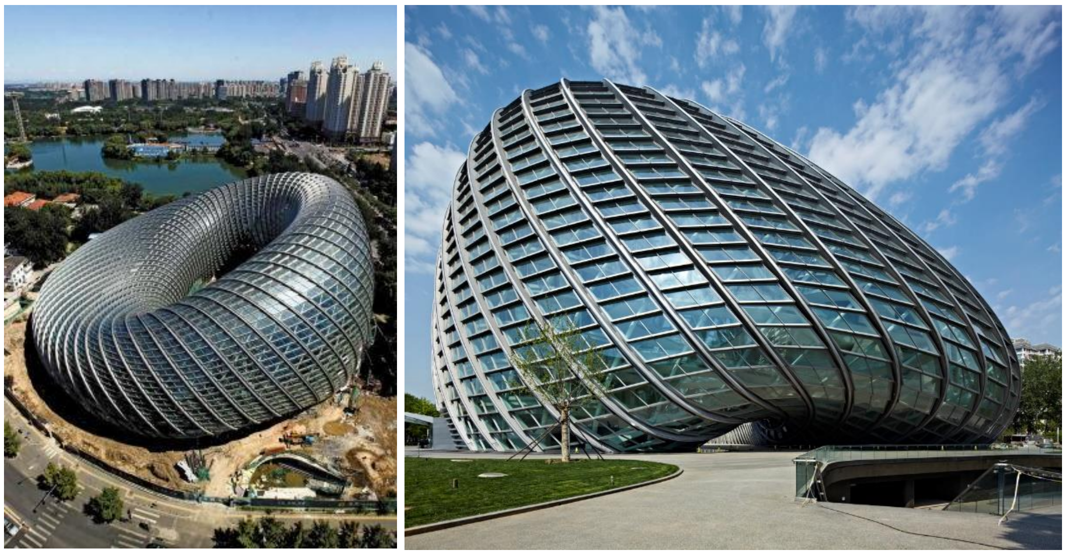


Figure 5.2 – Prototype object: Phoenix Internationa Media Center by BIAD Ufo, Beijing

This project required specific shape grammar determination. Computational system was used to control the model geometry. Meanwhile, BIM model was expoit for coordination and documentation purposes. As seen from the above, the project of Phoenix Internationa Center fits well the case study objectives covering full range of AIM specific tasks. The case study project is divided into eight stages:

1. Parametric Schema briefing
2. Decoding and reconstruction of Generative Grammar of Shape
3. Main Graph system modelling in Grasshopper
4. Connecting Graph with BIM authoring software, Revit. BIM model setup
5. Panelization & Optimization with Galapagos
6. OOS Model export from BIM to engineering software, Seismostruct.
7. DfMA Model export from Grasshopper to CAM software

5.2.1. Parametric Schema

The starting point of modelling Parametric model is inferring Parametric Schema of the object. Parametric Schema defines control parameters and determines parametric relationships laid at the basis of the parametric model. Visual observation and design documentation exploration led to the assumption of practical form-finding rules of the object. The building's sculptural shape prototype originates from the "Mobius Strip" which can be confined as a twisted torus with dynamic modifications of its sections including scaling, rotation and transforming. Furthermore, sections are subdivided and connected by structural nodes along the curves forming framework of facade structure. Control parameters include general form control parameters such as maximum attitude, torus radius, section semi-major and semi-minor axes. While maximum attitude and torus radius remain static, section axes are functionally defined and computed incrementally through the graph. Another group of parameters is related to section and structural framework control. In contains a number of sections, a number of subdivisions along section curve, shift number defining the configuration of connections between structural nodes, and offset distance between internal and external framework structures. The Parametric schema is presented below in Figure 5.3. Parameters are defined as:

<u>General group</u>

- maximum attitude – H_{max}
- torus radius - R
- section semi-major and semi-minor axes - R_{k1}, R_{k2}

<u>Section & Framework group</u>

- number of sections - K
- number of subdivisions along section curve - N
- shift number - i
- offset distance - O_{str}

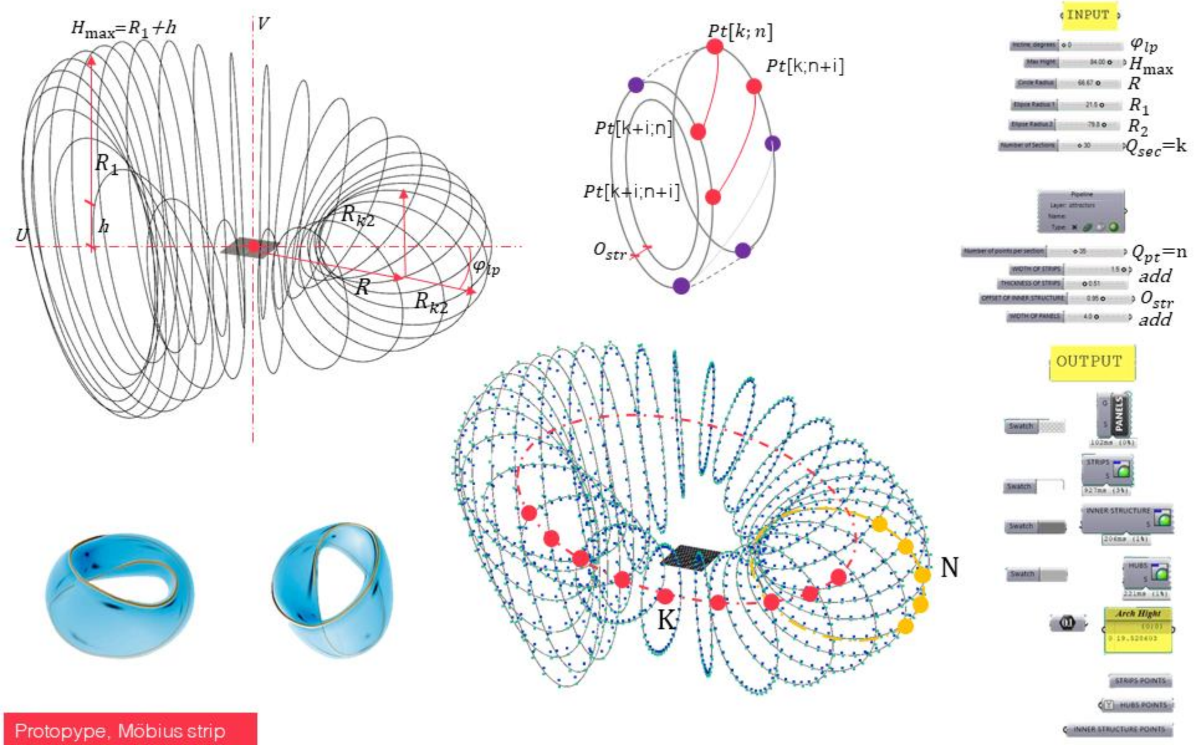


Figure 5.3 - Parametric Schema

5.2.2. Generative Grammar of Shape

After defining the Parametric Schema, the next objective is to arrange a rule set and transformative hierarchical order for the generation of the parametric model. For determination of this part of algorithmic modelling, the study proposes the terminology Generative Grammar of Shape (GGS). A GGS is inferred from the Phoenix International Media Center building based on visual investigation and design documentation exploration.

The term Shape Grammar (Stiny, 1972) may be described and considered on two levels – computational and spatial relationships. In computational theory, shape grammars are specific classes of rule-based expert systems in artificial intelligence which generate geometric shapes. The basis for rule composition is geometric transformation such as translation, rotation, reflection, and scale, which allow one shape to be a part of another sharing morphology. It can be used as an analysis tool, for breaking down of complex shapes and as a synthesis tool, generating complicated geometries starting from a basic shape. (Fasoulaki, 2008)

From the other hand, Shape Grammar represents the theory of visual, or even spatial, thinking. The term ‘shape grammars' more literally refers to visual design grammars. In other words, shape grammar is “the philosophy of looking at the world that is not through learnt or imposed decompositions, but through those that have a practical meaning at that point in time.” (Mine Özkar, 2008) Notably, the spatial aspect of shape grammars was crucial for its implementation in contemporary architectural theory and design framework for the last four decades. In the academic circles of architects, Shape Grammar was adopted even long before conventional drafting CAAD (Computer-aided architectural design) tools were developed. In particular, analytical grammars have been developed to describe and analyse architectural styles and historiographically relevant languages. (Figueiredo, 2019) Example of such Grammar is represented in Figure 5.4.

Usually, in practice, Shape Grammar is related to Generative Design as an autonomous generating system with iterative functions which allows the generation of multiple solution space.

However, in the context of the case study, it is regarded as a spatial break-down structure of the shape for generation algorithm reconstruction. Consequently, Generative Grammar of Shape is presented as an operational sequence with confinement to morphological rule sets within constraints defined by control parameters introduced in the Parametric Schema. Pointing out the nature of GGS as part of AIM methodology, the parametric model performs as a non-linear system allowing the control on different generation stages (levels) without sequential constraints in time. Therefore, the study proposes to use Levels instead of Steps in defining the operational sequence.

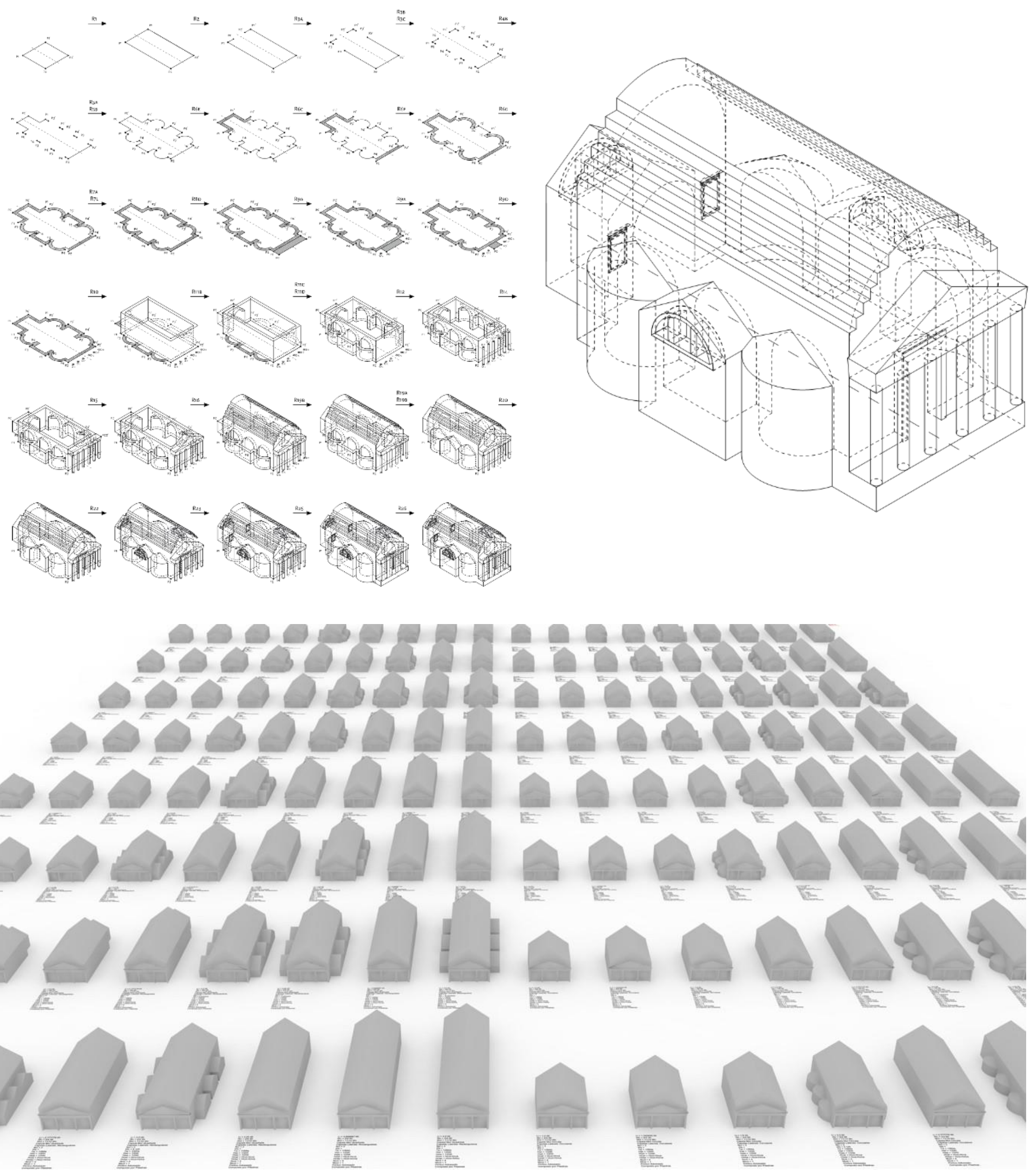


Figure 5.4 – Grammar of sacred rectangular plan buildings, as described in *De Re Aedificatoria* (Alberti L.B., 1485), derivation of a design solution(on top), solution space generated in Grasshopper (bottom), taken from lecture of Bruno Figueiredo within Master CourseworkBIM A+, 2019

On the first Level of GGS, loop transformation is scripted. Ellipsoid-like sections are rotated around a central point with functional dependency on Bezier graph, which controls sections in three dimensions: rotation, scaling, and displacement with attractors. (Figure 5.5)

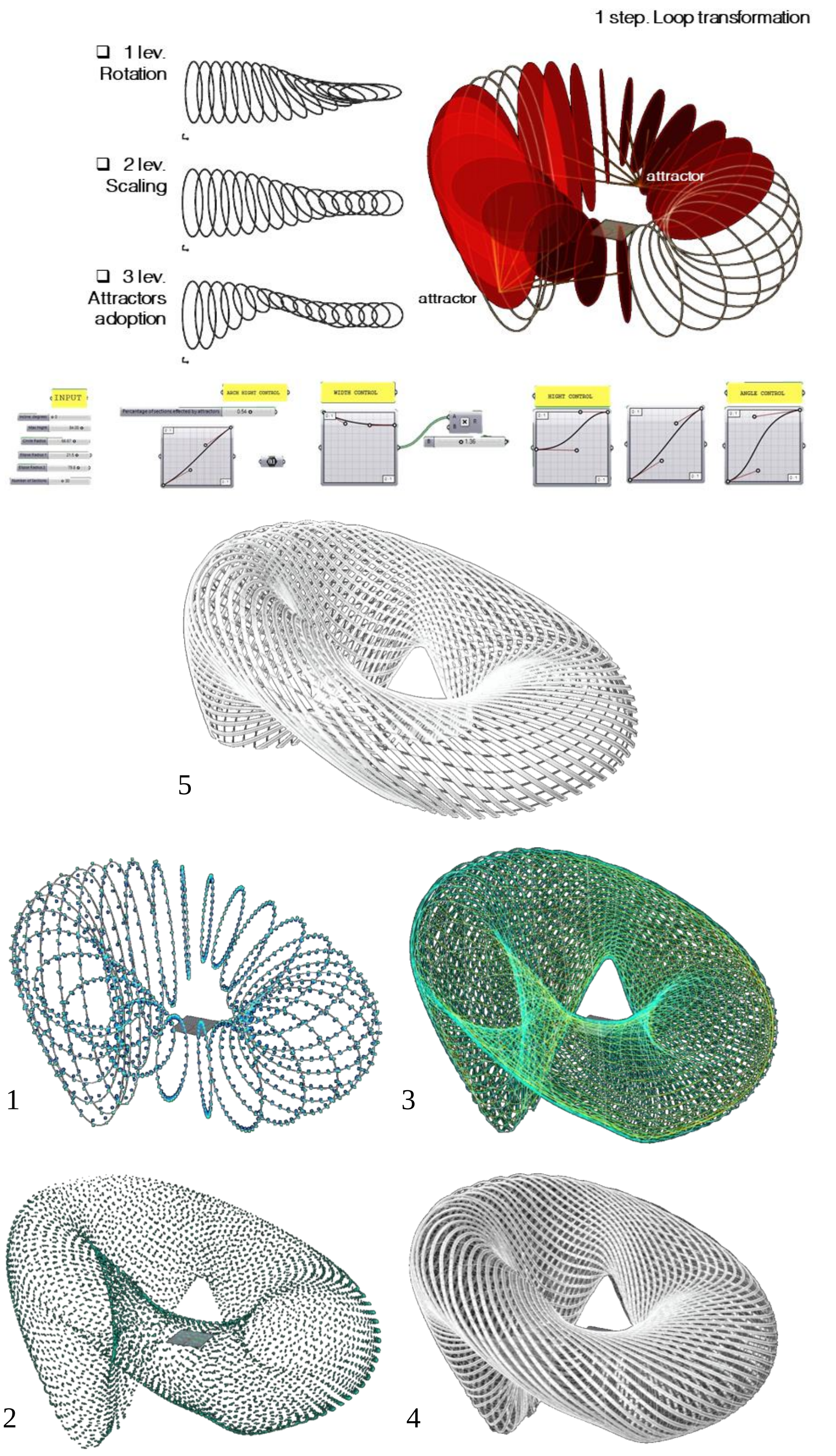


Figure 5.5 – Levels of GGS: 1st**Level (on top), 2nd Level (on bottom), generation sequence labled by numbers**

On the second Level of GGS, structural framework is generated. It consists of internal and external shells with the opposite direction of structural elements – inner trusses and outer bands. Thus, in the intersection points, nodes are placed connecting shells into a single space-truss framework. Furthermore, panels are located between bands finalising outer shell structure. For recreation of GGS, the next operational sequence is defined:

1. Sections are subdivided, forming two grids of points from inner and outer structures, which are then consolidated into a 2-branches list with setup order rules for joints
2. Nodes are determined in place of conjunctions between inner and outer structures
3. Outlines are created based on point grids and joint rules, and nodes are connected with lines
4. Solid geometry is created with outlines
5. A boolean operation is conducted for solid geometry to remove the lower part of the shape

5.2.3. Main Graph system modelling

In this section, according to Generative Grammar of Shape, the main Graph is scripted in Grasshopper GAE. Further, script definition is described in compliance with GGS Levels.

1st LEVEL: LOOP TRANSFORMATION

Two parallel flows form the first part of the script according to 1st Level of GGS. One flow generates a nominal shape basis and manipulates geometry. Meanwhile, the second flow transposes values for the shape transformation.

In the first flow, a loop of sections is generated along the circle with equally spaced perpendicular frames. Then sections are created with nominal ellipse shape. The number of frames and axes values of the nominal ellipse are controlled by input parameters.

In the second flow, the range of rotation angle values is generated based on the number of sections and maximum constant value - 90 degrees. Then values are remapped and transposed with Bezier graphs which are controlled in the input. Next, all sections are rotated with generated angle values and scaled with non-uniform factors in X and Y axes of frames created in the first flow. Scale factors are transposed with another Bezier graph from the same remapped numbers synchronizing transformations of the shape.

After the nominal shape is defined, local transformations should be applied to form gates to the courtyard. Thus, another range is created based on distance from central points of sections to attractors. Next, the range of distance values is filtered by input parameter, which defines a percentage of sections affected by attractors. Sections are filtered respectively. Then filtered values are remapped with one more Bezier graph controlling gate hight. After the remapping procedure, the range of values defines Z scale factor for affected sections. Finally, transposed values are applied scaling ellipses with basepoint in the heist extreme along the section. As a result, two gates are created, and sections merged again in the single loop. Loop creation and transformation is presented in Figure 5.6. Input parameters (Figure 5.7) are used according to Parametric schema (refer to the section 5.2.1 Parametric Schema) including Bezier graphs for manipulating transposal functions.

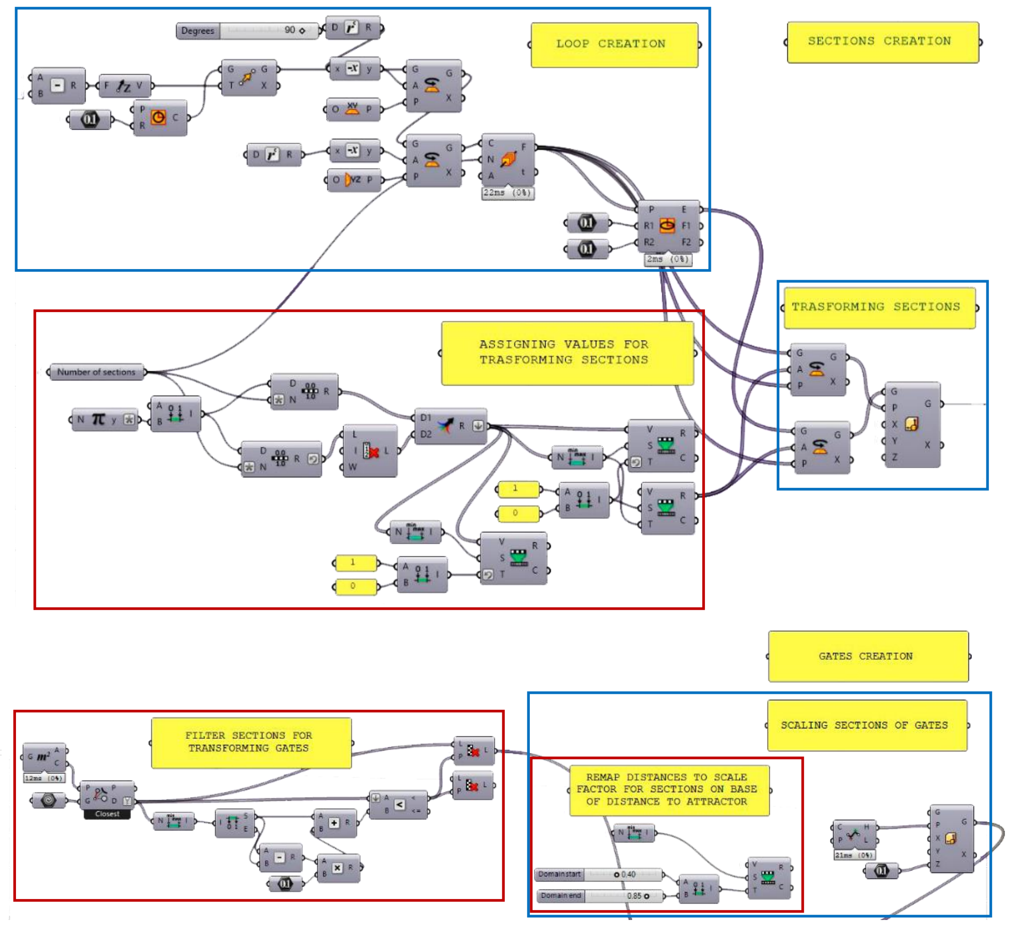

LOOP CREATION
SECTIONS CREATION
TRASFORMING SECTIONS
ASSIGNING VALUES FOR TRASFORMING SECTIONS
Number of sections
GATES CREATION
SCALING SECTIONS OF GATES
FILTER SECTIONS FOR TRANSFORMING GATES
REMAP DISTANCES TO SCALE FACTOR FOR SECTIONS ON BASE OF DISTANCE TO ATTRACTOR
Domain start
Domain end

Figure 5.6 - 1st Level part of Graph definition: Loop creation and section transforamations: 1st flow –nominal loop creation and geometry manipulation (blueframes), 2nd flow –values generating and transposal (red frames)

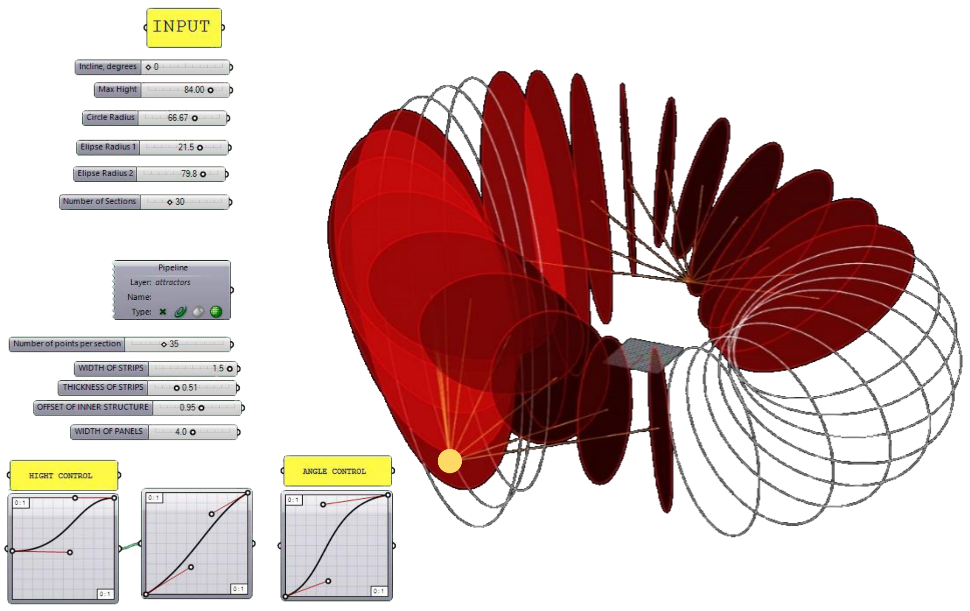


Figure 5.7 - Input control parameters (left), visual SOS (right)

2nd LEVEL: STRUCTURAL FRAMEWORK

In the second part of the Graph definition, Structural Framework is generated. On the first stage, grids of points are defined for inner and outer structures, starting from section curves division into equal-length segments. Then, sequences of joints are determined through retrieving relative item combo in the 2-branches data tree with an offset in each of list branch. As a result, rearranged lists of points are created. Next, polylines are generated, and control points are extracted for further interpolating curves reconstruction. Nodes are defined from common original points between inner and outer structures. The first stage is presented in Figure 5.8 with Graph definition part on bottom and System Output State on top.

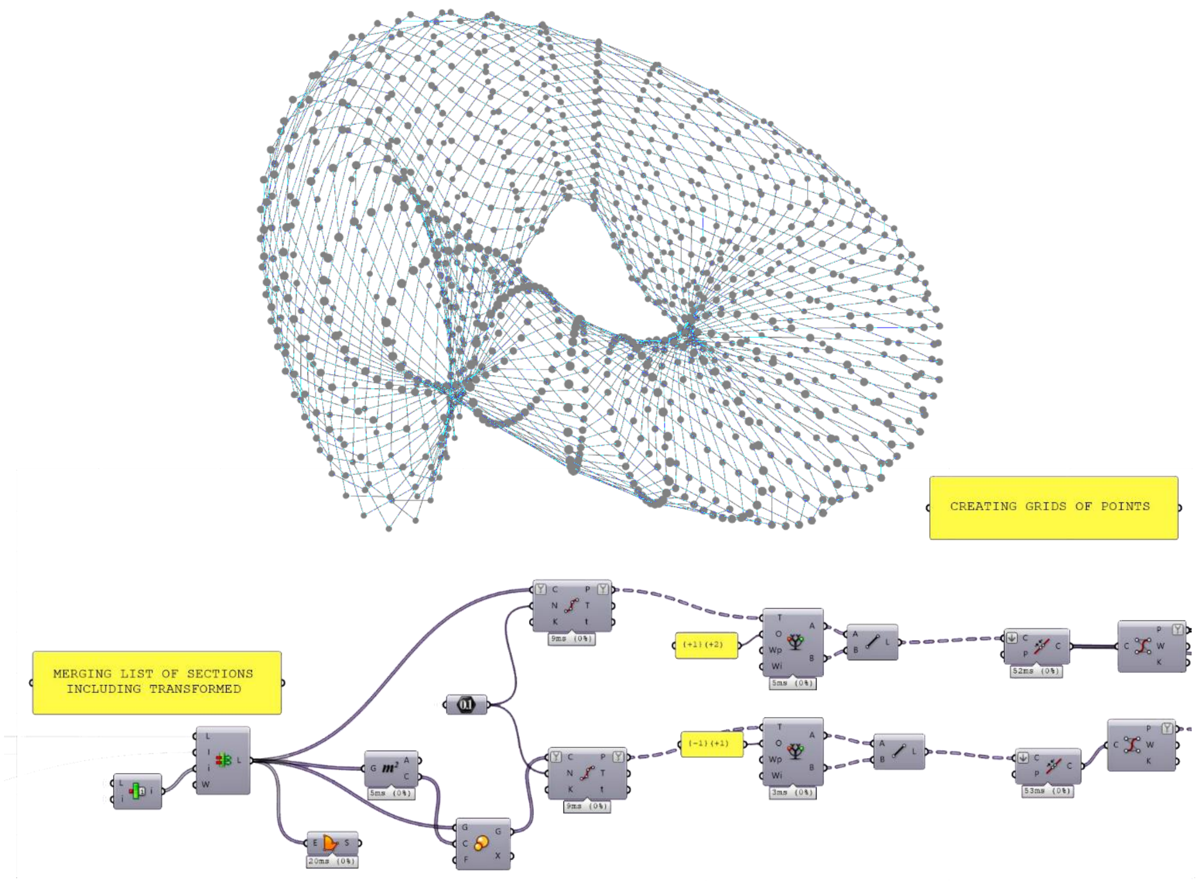
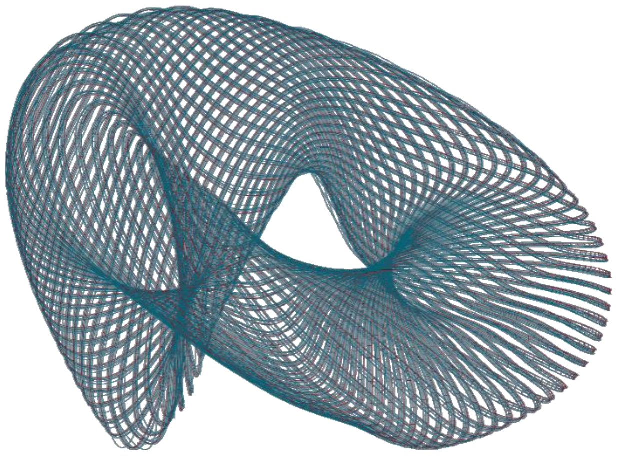
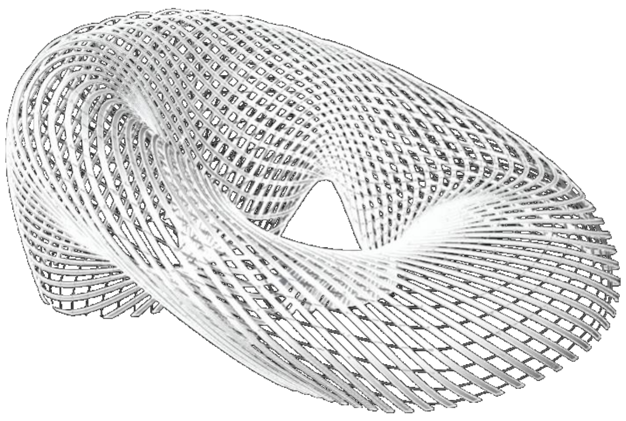


Figure 5.8 – 2nd part of definition, 1ststage: Point Grids definition part of Graph(bottom), visual SOS (on top)

On the second stage, curve outlines for modelling solid framework geometry are produced. The primary step before outline reconstruction is the determination of the base plane and axes extraction. It allows translating the point grids by local direction and building correct outlines afterwards. By this way, edge lines are generated based on moving point grids. Once all edges are created, solid geometry is built by loft. In the end, geometry trimmed by the Boolean operation.

As a result, the structural framework is created in the second part of the definition. The outer system is built from twisted bands around the shape. Panels are located between these bands. Meanwhile, the inner structure consists of pipes twisted in the opposite direction, with an offset from the former structure. Both structures are connected by nodes. Results are presented below in Figure 5.9.


A


B

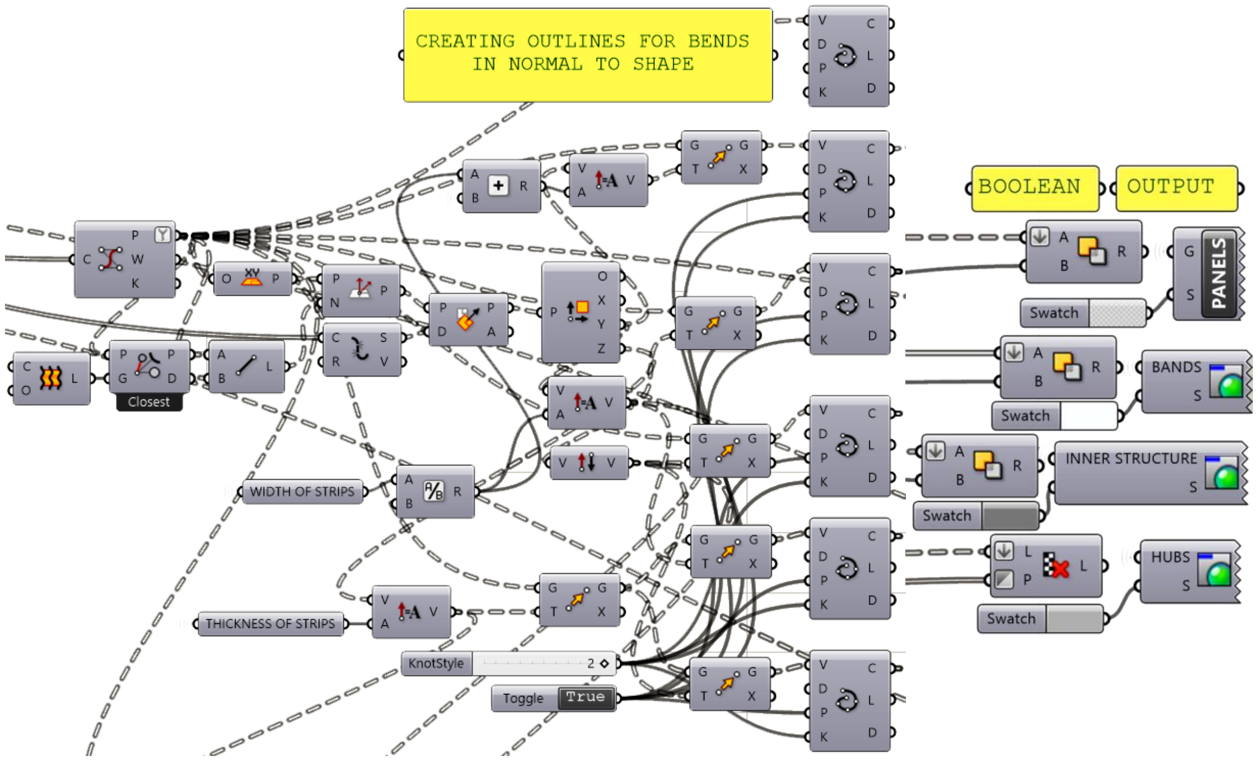


Figure 5.9 – 2nd part of definition, 2ndStage: creating outlinesof framework elements. Graph definition part creating outlines (on bottomleft), Output geometryin Graph (on bottom right), visual SOS upon outlines definition (on topleft), SOS upon Boolean procedure (on top right)

5.2.4. Connecting Graph with BIM

In this section, data exchange is arranged between AAM and BIM tools. During the project development process, it was discovered that the AIM project could be exchanged by using various approaches. According to Eastman, data exchanges between two software are typically carried out by using the following data exchange types (Eastman, 2008):

1. Direct, proprietary links between specific BIM tools

2. Proprietary file exchange formats, primarily dealing with geometry

3. Public product data model exchange formats

4. XML-based exchange formats

DIRECT AND INDIRECT DATA EXCHANGE TYPES

Data exchange types can be further categorized to Direct Data Exchanges (DDE) and Indirect Data Exchanges (IDE) (Janssen et al., 2015, p. 516). All the data exchange types mentioned above, except the first one, belong to IDE category.

Direct Data Exchange implies that objects are exchanged directly between GAE and BIM or analysis and simulations tools. To enable exchange, direct translators are called from one or both software. Thus, DDE processes rely on middle-ware tool interfacing capabilities and proprietory translators (Eastman, 2008). Notably, software vendors prefer to develop DDE as they can provide better support for keeping their customers within the ecosystem.

DDE correspond to the second Integration Level of AIM introduced in the present thesis (refer to the section 1.4, chapter 1, p. 18) Some of the available among DDE tools specific to AIM are Rhino.Inside, Speckle, and Grevit. As was mentioned in section 1.4, Rhino.Inside allows direct access to Grasshopper GAE and procedural system definition as well as to Rhino GUI. Procedural model is controlled within Grasshopper GAE right from Revit environment and translates this data interactively in both directions. Meanwhile, Speckle streams the procedural System Output States online through the cloud and allows to convert them to native objects in Revit automatically. Grevit sends the information in the same way as Speckle does but only in one direction from Rhino to Revit.

Indirect Data Exchange assumes a shared data schema used to link GAE and BIM or simulation tools. Indirect data exchange is based on an exchange format that is human readable. (Eastman, 2008) Levels of IDE vary depending of the amount of semantic information linked to the object. (Humppi, 2015)

On the first level, the type of data is confined by the geometry of the object. DXF, FBX, and SAT are examples of the first level file exchange formats. On the second level, the object contains metadata attached and describing its physical properties. The typical example of the second-level file exchange formats is an IFC standard. Besides, some plugins allow second-level file exchange as well between Grasshopper and Dynamo GAEs. Among them, some of the most ubiquitous tools are Rhynamo, HummingBird, and MantiShrimp.

Difference between DDE and IDE is expressed in several aspects. While Direct Data Exchange provides more fluent flow with less information loss, it also provides a risk of overloading system running several applications simultaneously. Another factor is the translatability of elements between AAM and BIM tools. Despite IDE exports data into external formats alien for both tools, the output can occur more reliable than DDE with the mutual support of these formats such as IFC or SAT due to their shared schema. Meanwhile, DDE depends on the development of proprietary translators for each software which should be synchronized. As a particular example, geometry quality can be affected with geometrical parts either occasionally lost or converted automatically into alien geometry type. The conversation of geometry is especially typical when overcomplicated shapes are translated. In this case, such shapes automatically are converted or divided into the type which is supported by recipient tool or format. Besides, Application Programming Interfaces also have constraints, which may lead to poor capability for the translators. (Humppi, 2015)

According to the mentioned above aspects, two approaches were tested. As it was estimated, DDE was heavy for hardware to handle. However, after a closer investigation of Rhino.Inside settings, there was an ability to turn off the preview mode which accelerated the data flow significantly. The other aspect showed up that geometry is better tackled with DDE in this particular comparison case. As a result, the DDE with Rhino.Inside plugin was selected for AIM integrated process.

AIM SYSTEM TRANSLATION TO THE BIM MODEL

The procedural model was translated to Revit through Rhino.Inside by generating floors, outer and inner structures and nodes. (Figure 5.10) First, outlines are extracted from the shape by intersections with planes defined by input altitude list. Then floor filter is applied to remove insufficient intersections through area mask. Next, Revit Floor object is created by outlines, floor type, and levels from altitude list. In parallel, outer, inner structures, and nodes are translated to direct shape objects of generic model type with defined materials from Revit default library. Each element is translated separately by grafting list. So, every element is a parametric instance and can be scheduled. The translated BIM model and initial AAM model with Graph are presented in Figure 5.11.

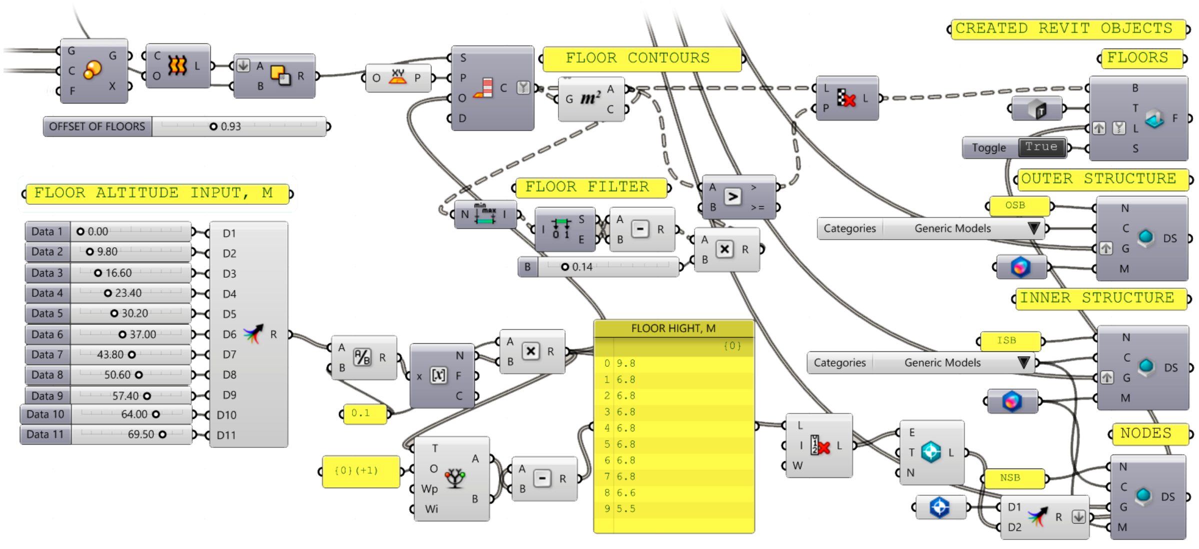


Figure 5.10 – Graph definition part of DDE setup to Revit

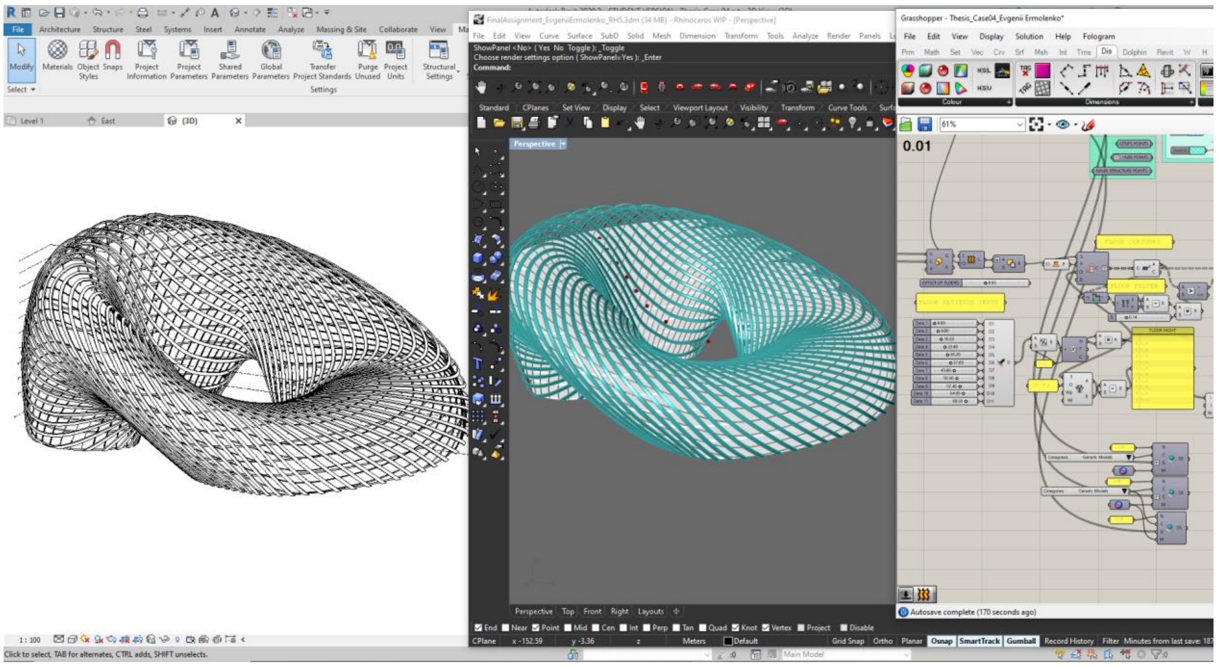


Figure 5.11 – DDE setup. From left to right: BIM model in Revit, AAM model in Rhino, procedural system (Graph) in Grasshopper

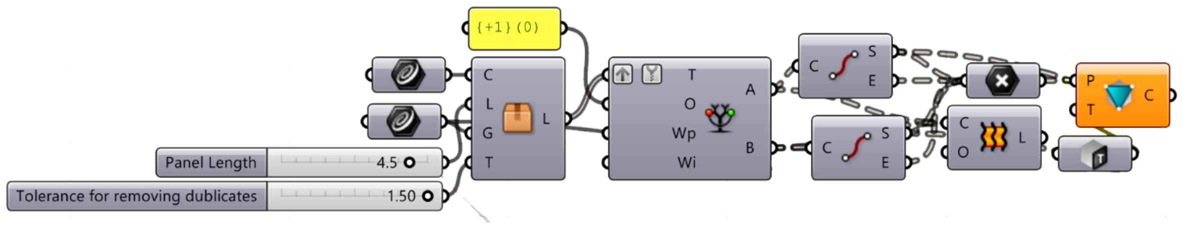
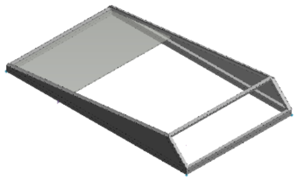


Figure 5.12 – Graph definition part creating adaptive panel components.



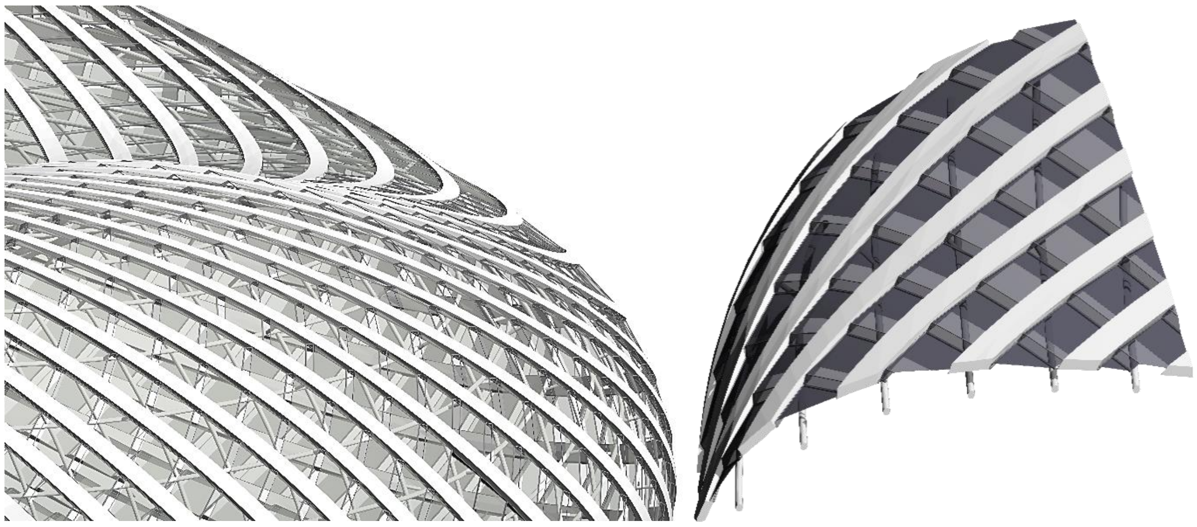


Figure 5.14 – Result panelized BIM model

5.2.5. Panelization & Optimization

In this section, panelization of the shell is considered. Once the structural framework is translated to BIM model, the panels should be determined between the bands. Their allocation is defined by four control points: couple on every neighbouring band edge. Thus, the adaptive component of the panel can be placed aligned to them. The adaptive component is created in Revit family editor, based on four adaptive points, respectively. The panel has incline and shade. (Figure 5.13)

Points are created by dividing curves with a defined input distance which is panel length. Then lists of points are coupled and connected by lines. Next, points are extracted again from the lines with rearranged order in a clockwise direction within each group of four points. Finally, grouped and sequential clockwise points are translated to adaptive points. Family of adaptive panel component is inserted into the BIM model according to sets of points directly from the Graph. The Graph definition is presented in Figure 5.12, and the result panelized BIM model is in Figure 5.14.

PANEL OPTIMIZATION

In the next stage, optimization of panels is performed. The goal of the optimization process is to achieve effective use of material and energy saving by adapting the shading system. Consequently, optimization is conducted in two phases: quantification and shading. Objectives, constraints, and variables define the optimization process.

In the quantification phase, the quantities of panels and types are optimized. Fitness function contains two objectives: the total quantity of panels and the total quantity of panel types respectively. Both objectives are evaluated toward minimization. When the amount of panels is insufficient, gaps can occur between the band and panel. Therefore, the water-tight model requires to constrain the maximum offset from the guide curve equalled to half of the band thickness. Variables are the length of the panels and point positions along the guide curves.

Three solvers are tested for optimization results: Evolutionary Solver, Simulated Annealing Solver, and Multi-Objective Evolutionary Solver. First and second solvers are contained in Galapagos plugin, the third solver is introduced by Octopus plugin for Grasshopper GAE. Subsequently, solution spaces are compared between solvers. As a result, Multi-Objective Evolutionary Solver in Octopus presents better-resolved outputs. One of the main advantages of Octopus compared to Galapagos is introduced Pareto-Principle for multiple goals. Solution space is visualized within the Cartesian Coordinate System. Each of axes can be aligned to a certain goal. Thus, it is easier to observe and extract solution that fits a particular goal with design priority defined. Solutions are presented in Figure 5.15. The best fitness values for types – 53, and panels – 3097. In the end, panels are updated in Revit.

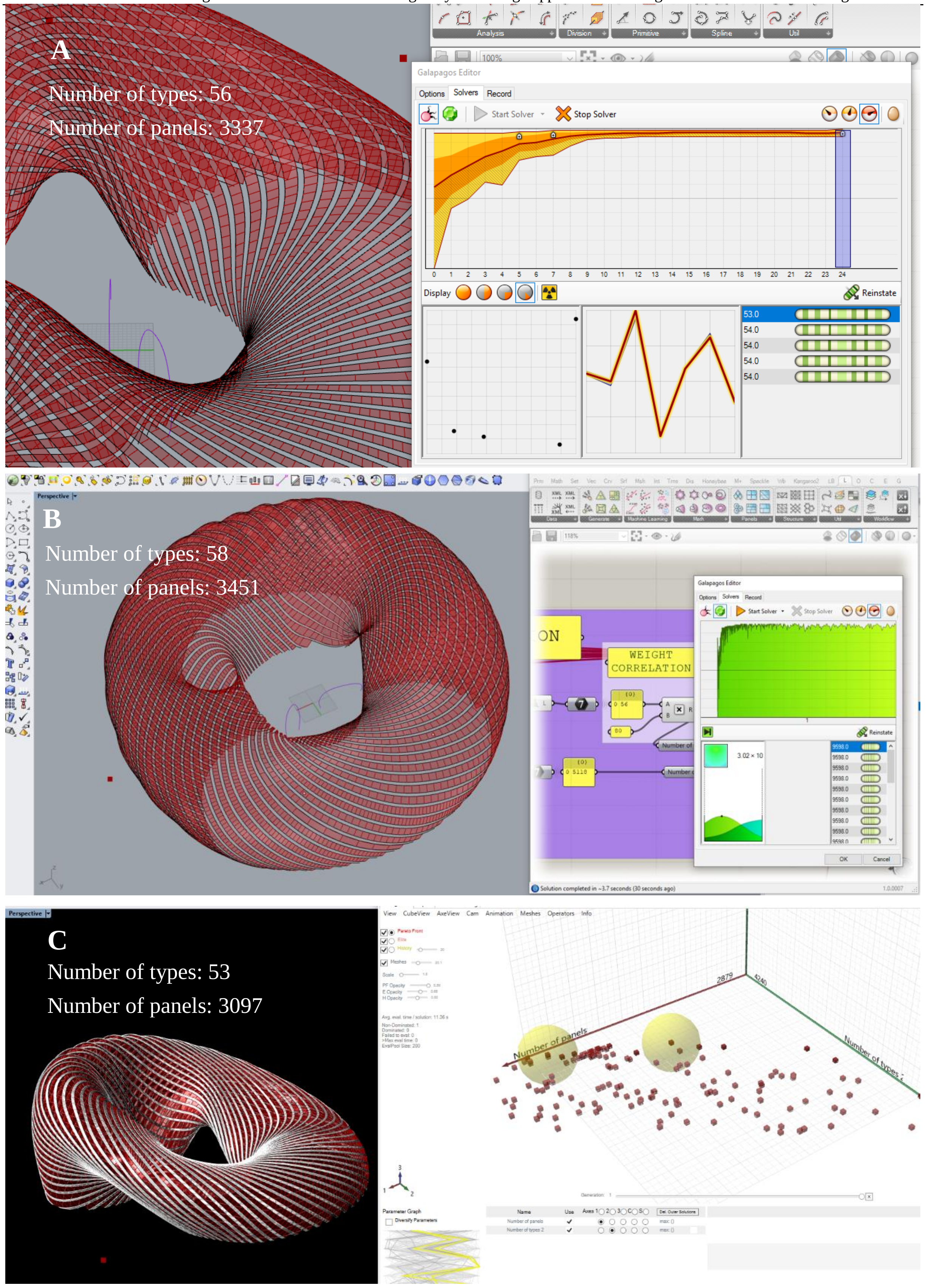


Figure 5.15 – Quantification Optimization process: Evolutionary Solver in Galapagos(Fig. A), Simulated Annealing Solverin Galapagos (Fig. B), and Multi-Objective EvolutionarySolverin Octopus (Fig. C)

After optimal panel quantity is found, the shading system is optimized as well to provide sufficient energy consumption for the building. The building is divided into two thermal zones: offices and public spaces. Subsequently, each zone corresponds to specific shading conditions which are objectives on the second phase of the optimization process. Thus, offices require less light and more cooling comparing to public spaces. Local adaptation of the shading system requires louvre opening control by input parameter. Corresponding instance parameter is created in adaptive panel component family in Revit. Sequentially, two different ranges of louvre opening are constrained: [0:0.95], [0:0.70]. The highest value - 1.00 – means full panel opening. The bigger range is defined for public spaces. The strategy applied to shading optimization does not exploit generative solvers intentionally as it utilizes Heuristic Method (see Glossary) implying single objective with predefined solution space. Instead, attractors with weights are used for computation of optimal solution. Each attractor corresponds to the thermal zone. Weight allows to adjust the volume of the thermal zone and consequently the influence on shading system. The Datascape (see Glossary) with Parametric Schema is presented in Figure 5.16.

This time, the Graph is scripted in Dynamo because once created panels in Revit loss their relation to point sets from the main Graph in Grasshopper. Therefore, assigning parameter values to adaptive components based on geometry created in grasshopper is not valid anymore. Hence, another Graph is scripted with input from adaptive panel components within Revit environment. (Figure 5.18)

First, the central point of the panel is extracted based on the bounding box. Next, two attractors are assembled according to thermal zones with weight property as a sphere with input radius value. Other input values control coordinates of attractor locations. Further, panels are grouped into two lists according to allocation between attractors corresponding to thermal zones. Hence, grouped panels are affected within defined shading range in the thermal zone. For grouping panels, a filter is used with mask determining the minimum distance value to attractor for every panel. In other words, the filter defines which attractor every panel is closer to. (Figure 5.18) Finally, the distances are remapped to values within the constrained range and set to "louvre opening" parameter of each panel family instance. The result BIM model with optimized shading system is presented in Figure 5.17.

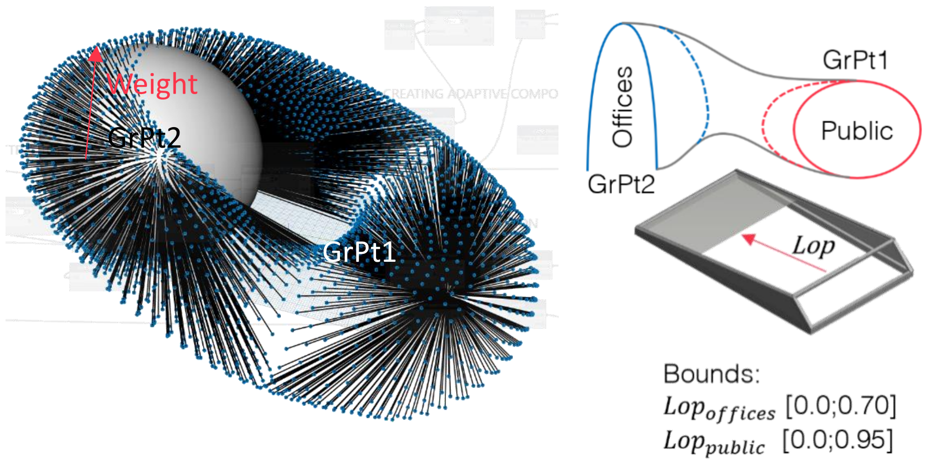


Figure 5.16 – Datascape of shading optimization: Grpt is group of panel points (left), parametric schema of shading system (right)

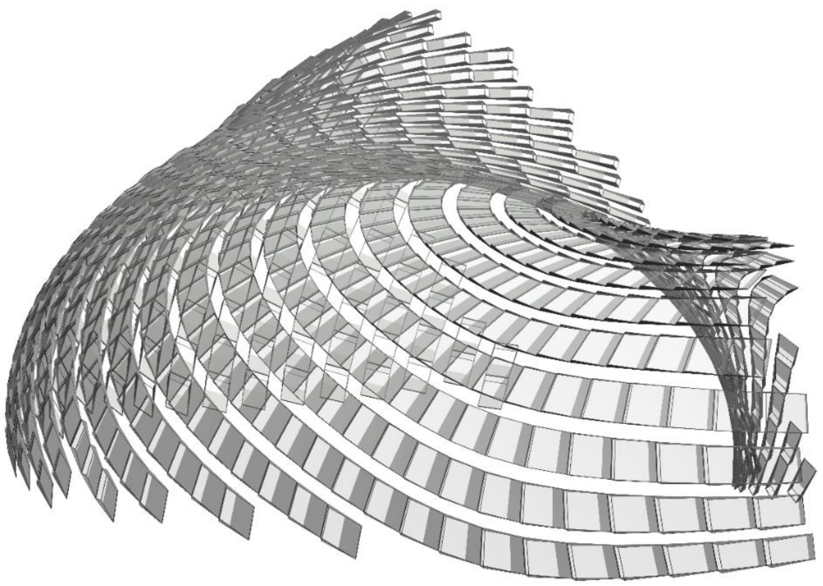


Figure 5.17 – The result sample part of BIM model in Revit after shading optimizationcomputed in Dynamo

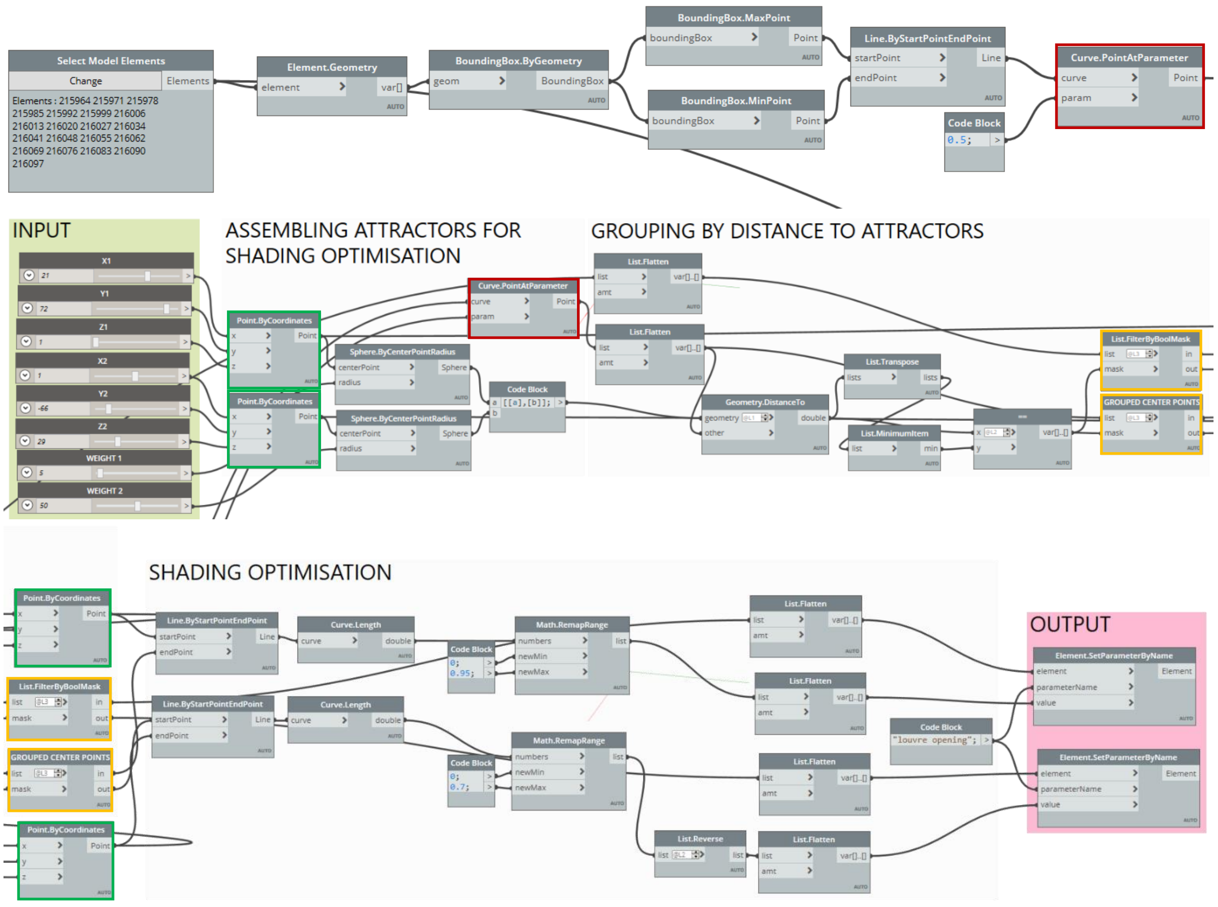


Figure 5.18 – Shading System Optimization Graph in Dynamo. Top-down: extracting panel central point, grouping by distance to attractors, shading optimization. Nodes connecting parts of script are framed in the same colour.

5.2.6. OOS Model export

There are a lot of research and practical case studies dedicated to interoperability issues within BIM. Once the model is imported in BIM software, it is enabled for the plenty of possible data exchanges within BIM environment, including both DDE and IDE (see Glossary). Industry Foundation Classes (IFC), an object-based open format for facilitating interoperability in BIM projects, allows exchanging data independently on vendor ecosystem. In its compliance, most of BIM software for design authoring, simulations, and coordination, nowadays support this format. While it would be easier to explore interoperability based on IFC implementation, the study focuses on Shared Mapping Data Exchange (SMDE) as the less investigated field.

Introduced by Janssen, Shared Mapping Process is a graphical mapping interface aimed to overcome limitations of DDE and IDE by allowing the users to define their own data mappers if others don’t exist. (Janssen, 2015) The basic type of mapping is called “Declarative Equivalency Mapping”, which is used for semantically equivalent data translations. More complex mappings are called “Procedural Query Mappings”, and tackled with the aid of graphical programming. (Janssen, 2015)

As an example of interoperability set with use of Shared Mapping between AAM and BIM tools, Object-Oriented Simulation engineering software, SeismoStruct is agreed with the partner company. SeismoStruct is a Finite Element package capable of predicting the large displacement behaviour of space frames under static or dynamic loading. (SeismoSoft, 2020) Seismostruct allows importing only XML files with a proprietary structure specific for this software. Meanwhile, the library of classes is very constraint. No DDE and IDE tools exist on the market supporting export BIM model to this software. Therefore, the only applicable alternative found is custom Shared Mapping Data Exchange elaborated in this study for space frame export. The Figure 5.19 presents that mapping between two data files involves three stages: parsing, mapping and serializing.

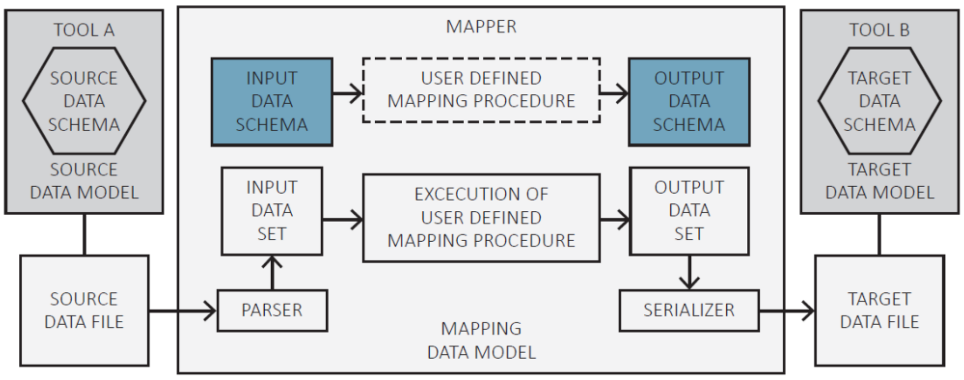
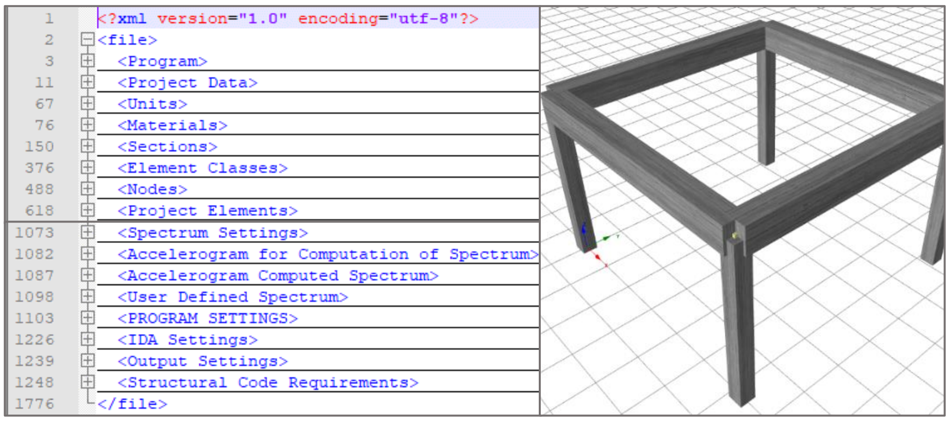


Figure 5.19 – Shared Mapping Process. The Figure is refered fromHarri Humppi’ Thesis (Humppi, 2015)

Elaboration of SMDE tool, first of all, requires a close investigation of the file structure. The sample file is explored for the structure in NodePad++. As seen in Figure 5.20, XML structure is quite logical but still is very long. With even the simplified sample model, it occupies 1776 lines of code. Thus, the decision is to explode the code by core elements, clean them up from the project data, and use them as a starting point fulfilling with all the necessary data from the BIM model. The Graph is scripted in Dynamo because it allows direct access to BIM elements with Revit API.

Primarily, the strategy is defined. According to the goal, stiffening core space frame of the office part is modelled as sample for export. Then in compliance with XML file structure, Graph is composed of modules corresponding to XML elements which supposed to be filled with object data. Among these modules are <Sections>, <Element_Classes>, <Nodes>, and <Project_Elements>. Finally, XML collector is developed which consolidates all the modules in single code exported further to XML file. Each module is scripted separately as a custom node. Subsequently, at the end of this section, a package of custom nodes is developed and utilized in the Graph.



```
<?xml version="1.0" encoding="utf-8"?>
<file>
  <Program>
  <Project Data>
  <Units>
  <Materials>
  <Sections>
  <Element Classes>
  <Nodes>
  <Project Elements>
  <Spectrum Settings>
  <Accelerogram for Computation of Spectrum>
  <Accelerogram Computed Spectrum>
  <User Defined Spectrum>
  <PROGRAM SETTINGS>
  <IDA Settings>
  <Output Settings>
  <Structural Code Requirements>
</file>
```

Figure 5.20 – Sample OOS model in Seismostruct:XML file structure (left), model in viewport (right)

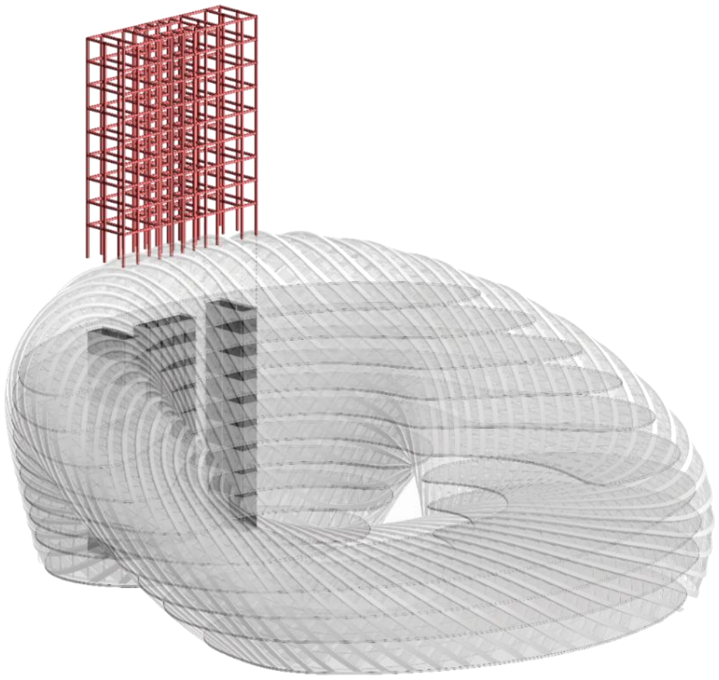


Figure 5.21–Space frame exported to Seismostruct from Revit: break-down scheme

First, Input group of parameters is determined. Input parameters include categories of structural framing and columns, the names of type parameters defining element dimensions, the export file path to an empty XML file which should be filled with code. (Figure 5.22) Next, Module <Sections> is defined which retrieves all the category types and creates its section XML elements based on inputs including Category and type dimension parameter names. (Figure5.24) It contains custome node with the corresponding part of source code which replaces its lines with extracted type parameter values. (Figure 5.25) Then Type Sections of beams and columns are combined into single code with the assigned number of section. Finally, Sections part is closed and ready to be translated into final code.

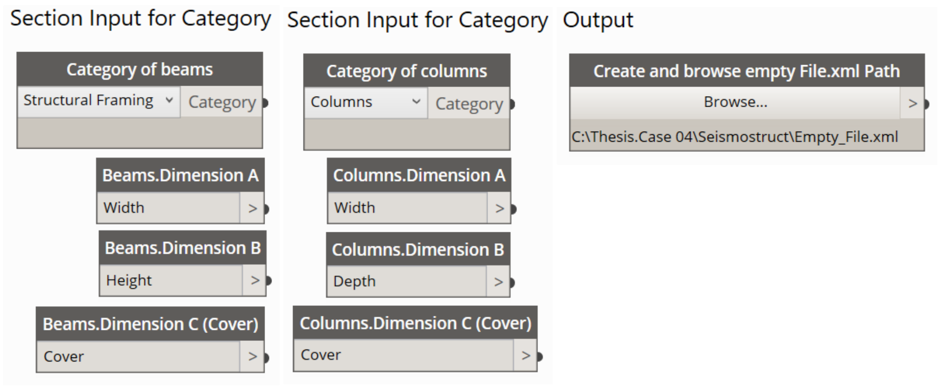
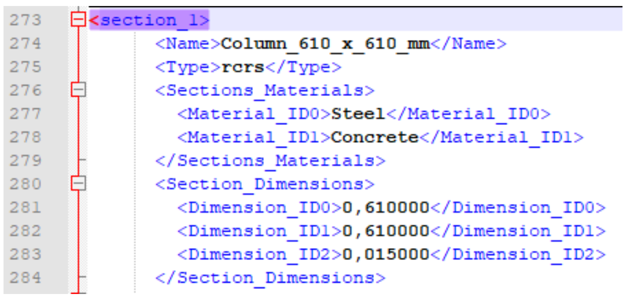


Figure 5.22 – Input parameters



```
<section_1>
      <Name>Column_610_x_610_mm</Name>
      <Type>rcrs</Type>
      <Sections_Materials>
        <Material_ID0>Steel</Material_ID0>
        <Material_ID1>Concrete</Material_ID1>
      </Sections_Materials>
      <Section_Dimensions>
        <Dimension_ID0>0,610000</Dimension_ID0>
        <Dimension_ID1>0,610000</Dimension_ID1>
        <Dimension_ID2>0,015000</Dimension_ID2>
      </Section_Dimensions>
```

Figure 5.23 – Part of XML code with section element

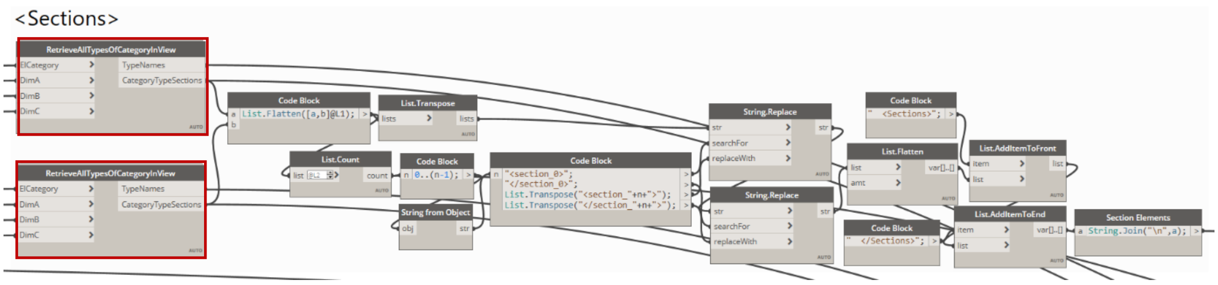


Figure 5.24 – Sections part of Graph definition: custom node containing Module 01is in red frame

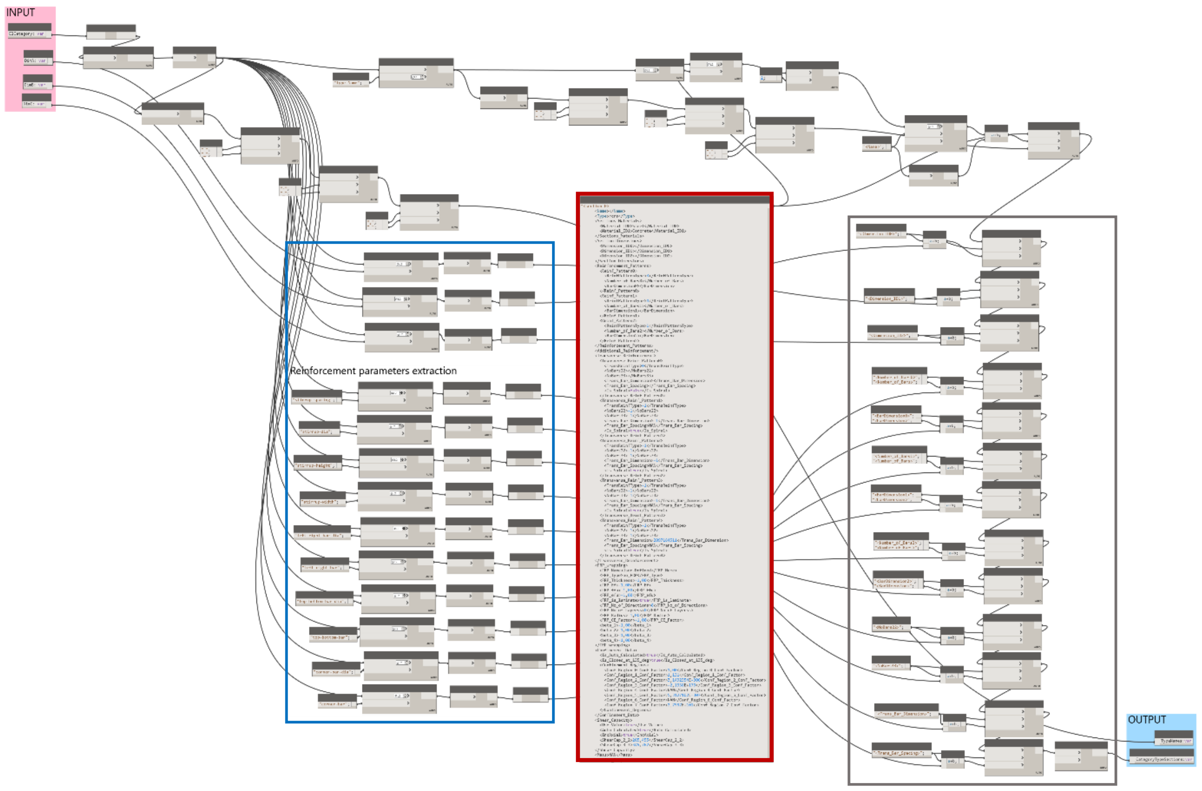
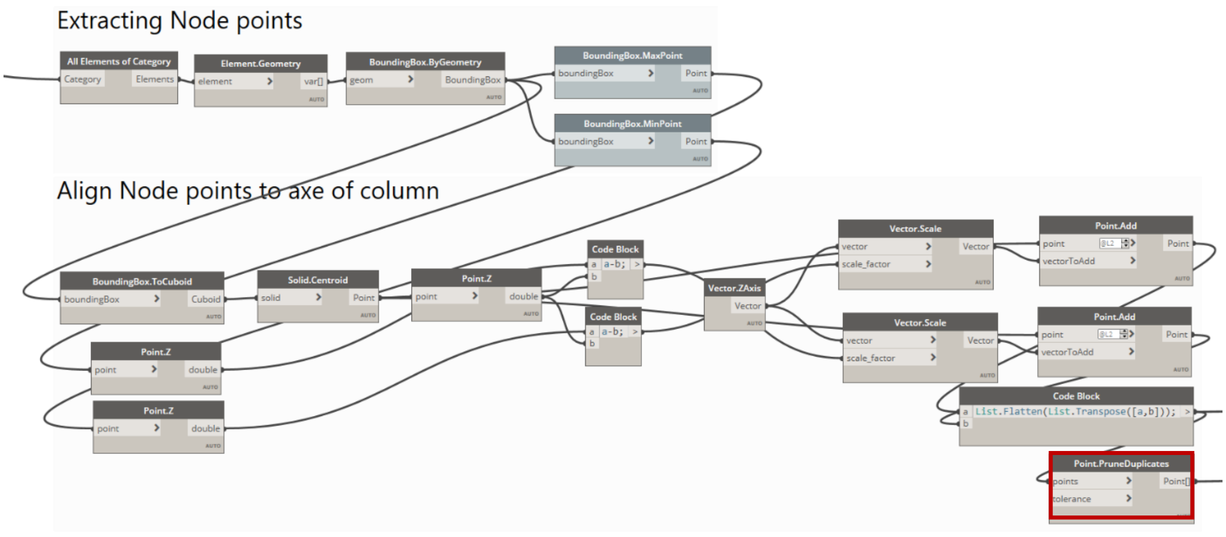


Figure 5.25 – Module 01: Type Sections by Category, extracting parameters in blue frame, XML source code in red frame, replacing lines of code in gray frame

Next step is the creation of Node Dictionary, which allows associating Nodes with elements by their coordinates. To begin with, Nodes are extracted as extreme points of columns from their bounding boxes. Then these points are aligned to column axes. Further, duplicates are pruned liquidating overlapping points from different elements with shared Node location. (Figure 5.26) After unique points are found, their coordinates are extracted, converted to strings, and joined. From joined coordinates tag, unique values are determined which become keys for Node Dictionary. Meanwhile, Node names are generated from the predefined prefix and key sequential numbers. Node names become values for Node Dictionary. Finally, Node Dictionary is composed and can be used further for associating with elements. (Figure 5.27)

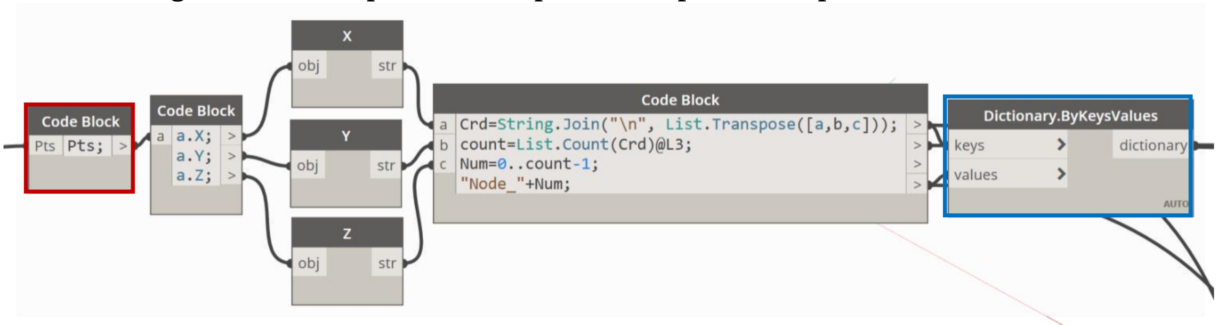
Once Nodes are determined in the Dictionary, they can be grouped by elements to associate element position. Notably, the same Node can be associated with different elements simultaneously. Therefore, spheres are created with the centre at the Node points. Then the filter is applied to Node points with a mask based on intersection events between element bounding boxes and spheres. Next, coordinates of grouped points are extracted, converted to strings, and joined. Finally, corresponding Node names are gathered from Node Dictionary by key values of grouped nodes coordinates. (Figure 5.28) The nodes of framework are presented in Figure 5.30.

As a result, all elements, including beams and columns, are associated with Nodes and can be further translated to the rest of XML elements with attached specific data requested by a certain line of code. By this way, other modules are created in custom nodes according to XML elements.



```
a-b;
List.Flatten(List.Transpose([a,b]));
```

Figure 5.26 – Graph definition part: Node points, Output node in red frame

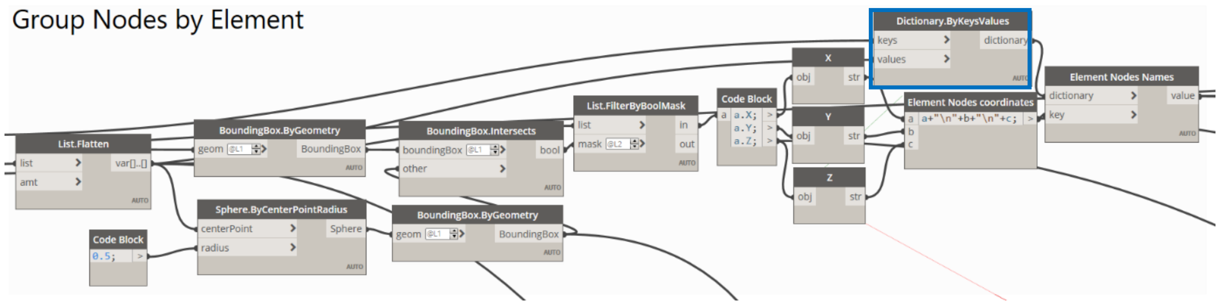


```
Pts;
```

```
a.X;
a.Y;
a.Z;
```

```
Crd=String.Join("\n", List.Transpose([a,b,c]));
count=List.Count(Crd)@L3;
Num=0..count-1;
"Node_"+Num;
```

Figure 5.27 – Graph definition part: Node Dictionary creation. Input node in red frame is the outputfrom Figure 5.26



```
0.5;
```

```
a.X;
a.Y;
a.Z;
```

```
a+"\n"+b+"\n"+c;
```

Figure 5.28 – Graph definition part: Element Nodes, Input node in blue frame is the output from the former figure

The last module in the Graph is XML collector (Module 06). It builds the whole XML code based on all other modules representing constituent elements of source code defined earlier. The package of custom nodes is created during the case study. Figure 5.29 shows the package nodes used in the Graph.

Eventually, full code is written into an empty XML file which further is imported to Seismostruct. The result of import is presented below in Figure 5.31.

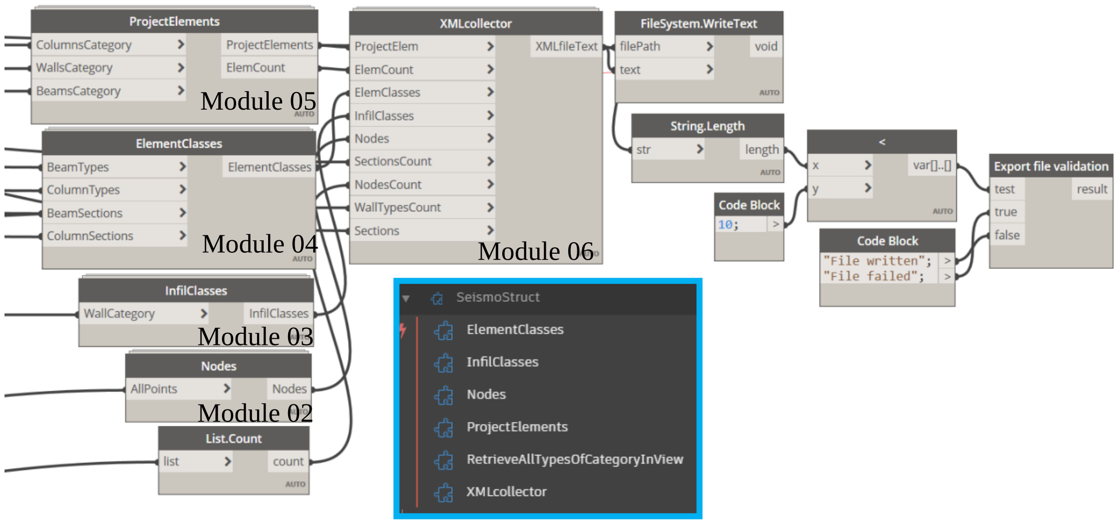


Figure 5.29 - Consolidation part of Graph definition writing XML file. Package of custom nodes used in the Graph is in blue frame

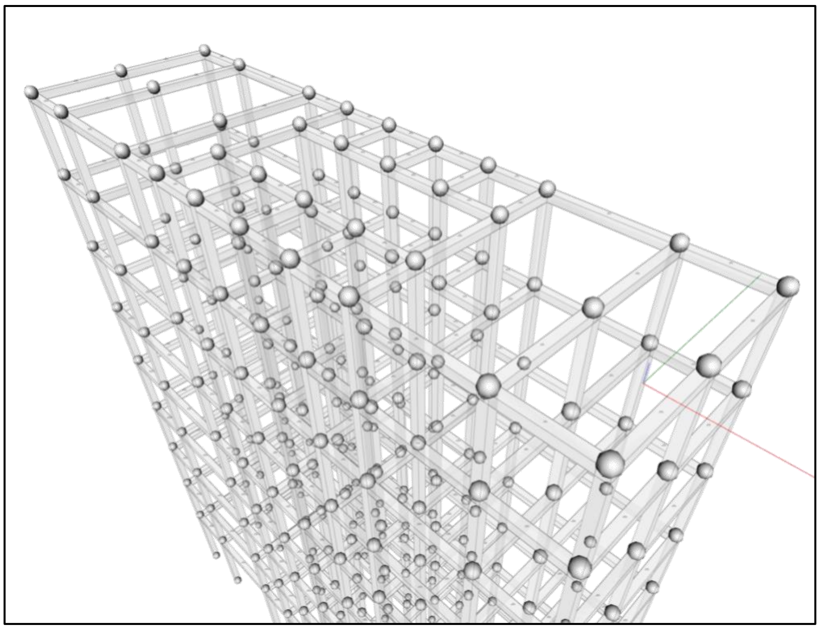
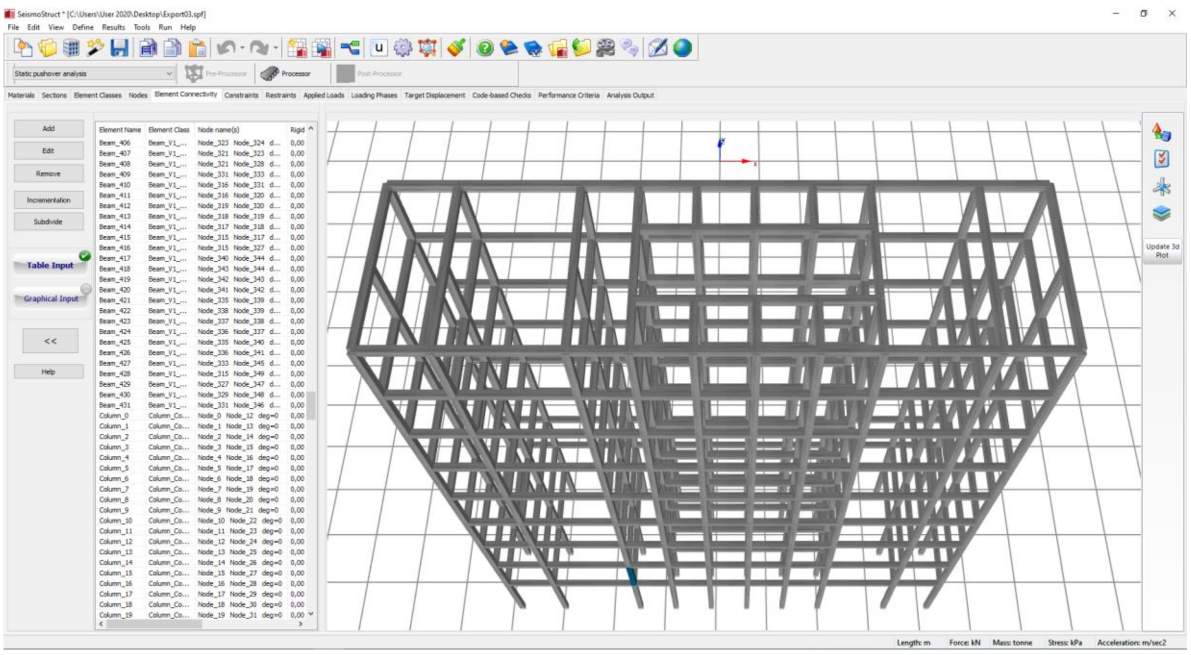


Figure 5.30 – Space frame in Dynamo viewport: Nodes are selected

In summary, the task possed in this section was one of the most challenging in this study. However, after careful investigation of the problem and breaking down XML structure by modules, the exporting tool was successfully written entirely in Dynamo GAE. Remarkably, modules scripted in custom nodes compute data slower than wholly written in the Graph, despite it reduces script length drastically.


SeismoStruct * [C:\Users\User 2020\Desktop\Export03.spf]
File Edit View Define Results Tools Run Help
Static pushover analysis
Pre-Processor
Processor
Post-Processor
Materials Sections Element Classes Nodes Element Connectivity Constraints Restraints Applied Loads Loading Phases Target Displacement Code-based Checks Performance Criteria Analysis Output
Add
Edit
Remove
Incrementation
Subdivide
Table Input
Graphical Input
<<
Help
Update 3d Plot
Length: m Force: kN Mass: tonne Stress: kPa Acceleration: m/sec2

Element Name	Element Class	Node name(s)	Rigid
Beam_406	Beam_V1_...	Node_323 Node_324 d...	0,00
Beam_407	Beam_V1_...	Node_321 Node_323 d...	0,00
Beam_408	Beam_V1_...	Node_321 Node_328 d...	0,00
Beam_409	Beam_V1_...	Node_331 Node_333 d...	0,00
Beam_410	Beam_V1_...	Node_316 Node_331 d...	0,00
Beam_411	Beam_V1_...	Node_316 Node_320 d...	0,00
Beam_412	Beam_V1_...	Node_319 Node_320 d...	0,00
Beam_413	Beam_V1_...	Node_318 Node_319 d...	0,00
Beam_414	Beam_V1_...	Node_317 Node_318 d...	0,00
Beam_415	Beam_V1_...	Node_315 Node_317 d...	0,00
Beam_416	Beam_V1_...	Node_315 Node_327 d...	0,00
Beam_417	Beam_V1_...	Node_340 Node_344 d...	0,00
Beam_418	Beam_V1_...	Node_343 Node_344 d...	0,00
Beam_419	Beam_V1_...	Node_342 Node_343 d...	0,00
Beam_420	Beam_V1_...	Node_341 Node_342 d...	0,00
Beam_421	Beam_V1_...	Node_335 Node_339 d...	0,00
Beam_422	Beam_V1_...	Node_338 Node_339 d...	0,00
Beam_423	Beam_V1_...	Node_337 Node_338 d...	0,00
Beam_424	Beam_V1_...	Node_336 Node_337 d...	0,00
Beam_425	Beam_V1_...	Node_335 Node_340 d...	0,00
Beam_426	Beam_V1_...	Node_336 Node_341 d...	0,00
Beam_427	Beam_V1_...	Node_333 Node_345 d...	0,00
Beam_428	Beam_V1_...	Node_315 Node_349 d...	0,00
Beam_429	Beam_V1_...	Node_327 Node_347 d...	0,00
Beam_430	Beam_V1_...	Node_329 Node_348 d...	0,00
Beam_431	Beam_V1_...	Node_331 Node_346 d...	0,00
Column_0	Column_Co...	Node_0 Node_12 deg=0	0,00
Column_1	Column_Co...	Node_1 Node_13 deg=0	0,00
Column_2	Column_Co...	Node_2 Node_14 deg=0	0,00
Column_3	Column_Co...	Node_3 Node_15 deg=0	0,00
Column_4	Column_Co...	Node_4 Node_16 deg=0	0,00
Column_5	Column_Co...	Node_5 Node_17 deg=0	0,00
Column_6	Column_Co...	Node_6 Node_18 deg=0	0,00
Column_7	Column_Co...	Node_7 Node_19 deg=0	0,00
Column_8	Column_Co...	Node_8 Node_20 deg=0	0,00
Column_9	Column_Co...	Node_9 Node_21 deg=0	0,00
Column_10	Column_Co...	Node_10 Node_22 deg=0	0,00
Column_11	Column_Co...	Node_11 Node_23 deg=0	0,00
Column_12	Column_Co...	Node_12 Node_24 deg=0	0,00
Column_13	Column_Co...	Node_13 Node_25 deg=0	0,00
Column_14	Column_Co...	Node_14 Node_26 deg=0	0,00
Column_15	Column_Co...	Node_15 Node_27 deg=0	0,00
Column_16	Column_Co...	Node_16 Node_28 deg=0	0,00
Column_17	Column_Co...	Node_17 Node_29 deg=0	0,00
Column_18	Column_Co...	Node_18 Node_30 deg=0	0,00
Column_19	Column_Co...	Node_19 Node_31 deg=0	0,00

Figure 5.31 – Imported XML file in Seismostruct viewport, elementproperties are shown in Element Connectivity tab on the left

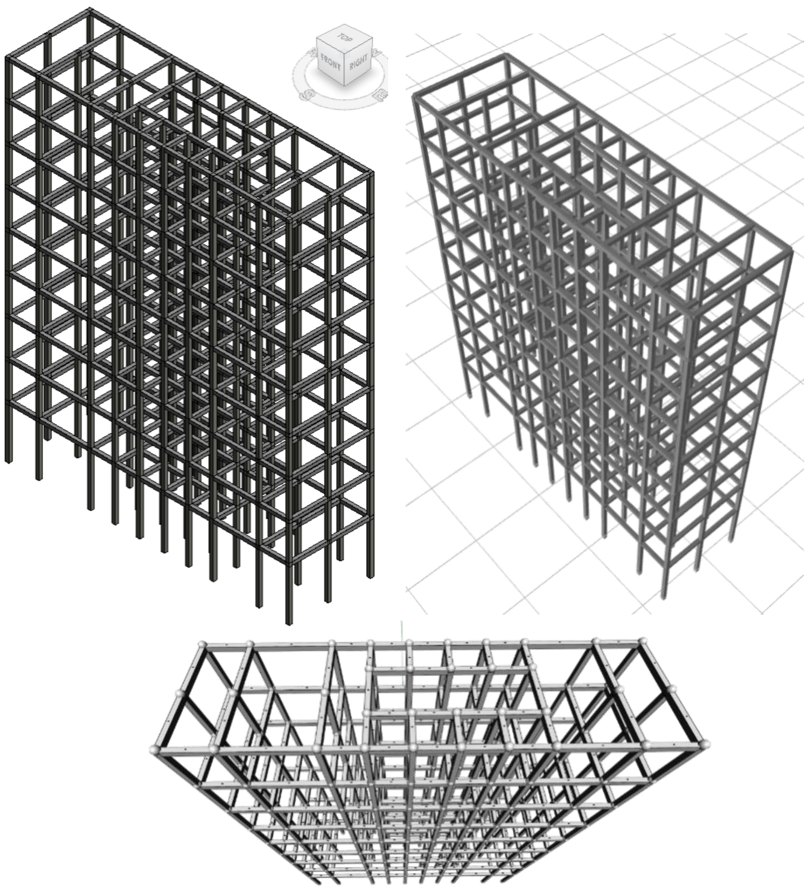


Figure 5.32 – Space frame models: Revit model (upper left), Seismostruct model (upper right), Dynamo model (bottom)

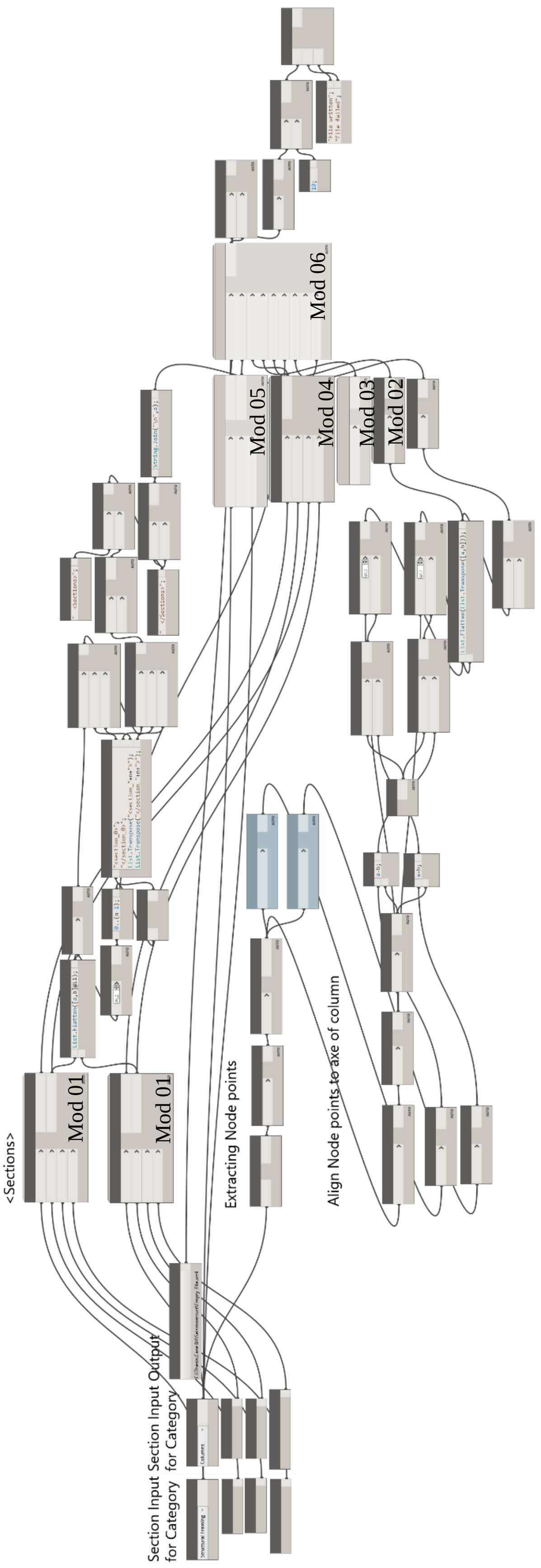

Mod 06
Mod 05
Mod 04
Mod 03
Mod 02
<Sections>
Mod 01
Mod 01
Extracting Node points
Align Node points to axe of column
Section Input for Category
Section Input for Category
Output

Figure 5.33 – Full Graph definition of XML export in Dynamo

5.2.7. DfMA Model export

Referencing to the chapter 4, Design for Manufacturing and Assembly assumes integration of material and production information into parametric model which affects the design process. Thus, such a model becomes manufacturing aware and supports lean design process from abstract to realization. In other words, DfMA enables the identification, quantification and elimination of waste or inefficiency in product manufacture and assembly. (ICE, 2020)

The simplified mockup model is elaborated in this section for 3d printing test. The model is entirely controlled by main Graph scripted in Grasshopper GAE. Prepared geometry is baked to Rhinoceros and then exported to Cura LulzBot, CAM software generating g-code, considering the constrains of the 3d printer, and the manufacturing process - Fused Deposition Modelling (FDM), and directly sending model to the 3d printer.

Following main principles of DfMA (see section 4.1.3), during the preparation, the geometry of parts is simplified and unnecessary features are avoided. Tolerance of the model set up sufficient for the particular task. Sucequently, structural framework is simplified: the single generalized shape is used instead of panels, bands are rounded and extruded to allow filament to pass through, the thickness of the shell is increased, and the basis is created in the ground plane for support. The prepared model is baked into Rhinoceros. The result model is presented in Figure 5.34. Next, model is exported in STL format and opened in Cura LulzBot software. This software splits the geometry into layers and allows advanced setup of 3d printing process including quality settings (outer, inner wall line width), infil dencity and pattern, print speed, building plate adhesion and generating support. After model setup in Cura LulzBot the process took around two hours to print the mockup model. The result is in Figure 5.35.

In conclusion, the DfMA is well-established modelling process within Digital Design techniques. Though, it requires special knowledge of material behavior and equipment as well as experience to achieve high quality of manufacturing adopting model for production.

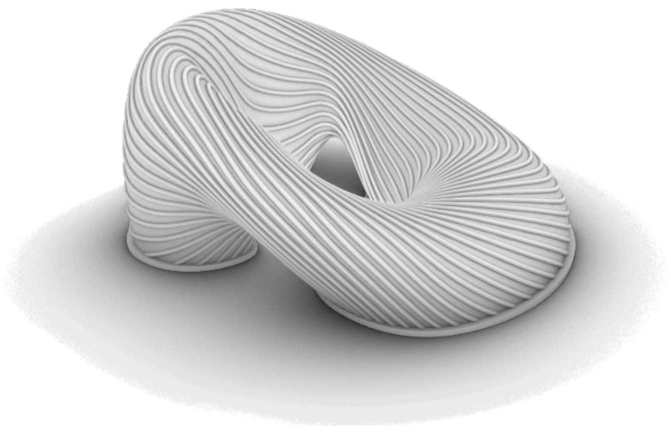


Figure 5.34 – Mockup model in Rhinoceros

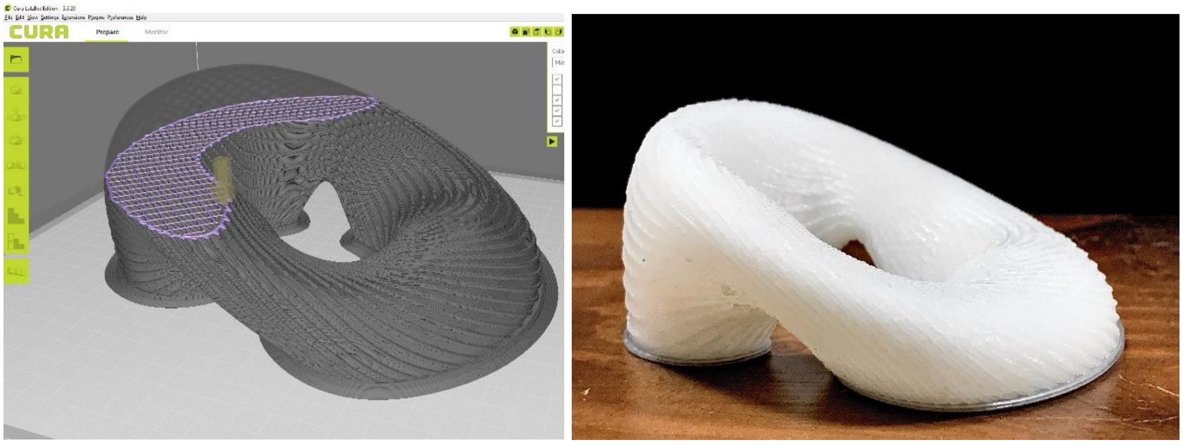


Figure 5.35 – BfMA model in Cura LulzBot(left), 3dprinted mockup model (right)

5.3. CONCLUSION

In conclusion, the case study consolidates all the efforts and experience acquired during former Case studies. The AIM methodology is tested within the most primer aspects such as procedural modelling, interoperability, object generation, design optimization, and manufacturing. As a result, AIM system is built. It enables accurate and diverse data flows under specific discipline needs while controlled from the central model by Graph in Grasshopper. Moreover, the AIM system enables interactive Direct Data Exchange while allows scripting custom Shared Mapping Data Exchange tools. Advantages of both AAM and BIM methodologies are combined within the AIM framework.

Noteworthy, the modelling process in AIM is perceived as brain-twister rather than routine. Thus, the puzzling design problem encourages the designer to think systematically with clearly defined goals and means. Subsequently, AIM reveals great computation opportunities along with object-oriented design advantages.

In a nutshell, the study shows that digital design methodologies are hard to investigate exceptionally through literature review and exploring theory only. The practical modelling helped the researcher to systemize knowledge gathered before and to achieve a comprehensive vision of the subject overall.

This page is intentionally left blank

6. SUMMARY

I consider this study as vitally important both in terms of contribution to the development of AIM and personal achievements in the subject investigation. The study has been a challenging journey for me. Except for the study value as research investment, it matters for me as a tool that I have used to gather knowledge. I believe that this comprehensive approach to investigation will be a fruitful and helpful for other researchers in the future as well.

As the research has shown, the studying process has contained theoretical and practical work. The study proceeded in an incremental cyclic manner from the most general theoretical part toward particular topics and case studies. Agile methodology was implemented in research. Upon transition to the next case study chapter, theory and literature were revised through the lens of the subject. Hence, the objectives were clarified in the introduction of each sequential case study chapter. By this way, clearly defined objectives from sufficient points of view established methods and tools considered in the study. Scheurer F. once said - “tools don’t find solutions, solutions produce tools”. (Scheurer, 2013) The present research confirms that statement.

Based on both practical and theoretical approaches, the conceptual framework has been discovered throughout the research. The framework contains taxonomy from related terminology tightened from the perspective of Algorithm-aided Information Modelling. It was an intricate part of the research as most of the terminology has had ambiguous definitions so far. Therefore, the study necessitated mapping taxonomy proposal according to the subject of research.

Another part was dedicated to case studies where diversified methods and tools were elaborated from scratch to final solutions readily for implementation in different work stages of the architectural studio. The case studies itself encouraging but became a significant challenge in practice. Nonetheless, thanks to this experience, knowledge of the subject was enhanced, and professional satisfaction was acquired.

In the following sections, I shortly summarize the primary points of all principal parts according to research methodology and extend investigation with AID Theoretical Synthesis.

6.1. THEORY BASICS: CONCEPTUAL FRAMEWORK

The research began with theory investigation. Primarily the main concepts assumed essential for the subject were explored. Object-Oriented Design and Algorithm-Aided Design were investigated as part of CAD history. It was discovered that both concepts have a long historical track starting even before computerization. However, while CAD tools have been mostly used as an assisting means for technical drawing, digital design, is still considered immature. Notably, OOD and AAD, as presentative methodologies of digital design, were developing in parallel for decades. Although while the former one was used for standardized project development, the latter was mostly exploited for experimental form-finding. Potential advantages of possible collaboration were hidden for a while.

However, the last decade has shown rapid ascendance of more user-friendly design software with open API. Meanwhile, easier associative programming languages such as DesignScript engaged designers to program their concepts and automate design processes. Subsequently, a growing community of scripters appeared. This influence initiated a new generation of specialists ready to make their innovative contribution to the development of a digital design. Such fast-paced alterations caused me to investigate existing CAD approaches for effectiveness, speculate about the emerging methodology and examine it within present-day realities.

During the literature review, it was affirmed that are still numerous approaches that can reform digital design. The present research proposes to generalize these approaches as Algorithm-aided Information Modelling. Meanwhile, AIM leaves a great variety of workflows within the unified framework based on well-defined taxonomy.

6.2. CASE STUDIES

In Introduction to each Case Study, it was investigated how AIM methods and tools can be used and how AIM can facilitate other design methods through the lens of the particular case. It included studying of existing primers, workflows, and arising under their potential implementation issues. Consequently, an analysis was conducted, and strategies for case practical part were planned. Among commonly identified issues there were:

- determination of main actors and parties involved in the workflow and their relations
- goal-specific tool planning to avoid data and time loss
- interoperability degree between tools to achieve fluent design processes.

Then, according to strategies defined in Introduction to Case Studies, workflows and tools were practically elaborated. The intent of the present research in this part was to examine the application of the proposed methodology with particular focus on the internal design process.

In this fashion, the description style was a primer with incremental process disclosure by milestones to let the reader immersive exploration. I believe that such an approach is helpful for the researcher as well as for the reader to identify advantages and possible issues inside-out. It will also contribute to a further critical investigation of the subject.

In Case Studies, the researcher focused on implementing AIM methods and tools to modify and improve existing workflows within OOD as considered a primary relevant problem on a way of transition toward new design methodology. In the final case, the AIM system was elaborated from scratch to demonstrate the main advantages of such an approach and testify complete AID workflow.

Within design processes described during Case Studies, such aspects typical to AID methodology as model & documentation generation, design automation, interoperability, manufacturing control, performance analysis and optimization were considered. Except the advantages of these aspects, the limitations were identified as well.

The first limitation is related to the usage of user-defined objects in GAE. User-defined objects created and controlled by script are non-editable in BIM environment meanwhile. Thus, first they need translation or “baking” from GAE to BIM software UI whereafter objects loose control by script. Subsequently, GAE integrated to the BIM software currently unable to allow visual scripting and manual modelling to happen simultaneously for the same objects. In AIM scripts could control the overall shape of the model while manual modifications could be made in BIM software. The challenge is to find the right ways to treat manual alterations if the script modifies the model. All in all, the possibilities of combining manual modelling and visual scripting should be further developed. Besides, user-defined objects may be hard to export to third-party tools as the objects and their geometry type are not always supported by IFC.

Another limitation is the lack of precise system joints. This issue caused by object-standards such IFC which objects are based on. Thus, objects will always contain limitations. Particularly, standard objects are usually based on a Cartesian coordinate system. Thus, the problem is that these objects don’t behave precisely when they are used in the other coordinate systems that are commonly used in scripts. Therefore, some manual verification and adoption of joints between objects are still requred.

Next limitation is the exclusivity of GAEs, which are non-interoperable with each other. In other words, parametric systems function within hosting GAEs. Overcoming this limitation requires Indirect Data Exchange or Shared Mapping process initialization. As observed in section 5.2.6, Shared Mapping process proposed by Janssen in 2015 is the graphical mapping interface as a new way to exchange model data. (Janssen, 2015) This approach allows the users to define their own data mappers if direct or indirect processes don’t exist.

Finally found limitation is related to simulations tools. Usually, heavy computations typical for simulation processes interrupt dynamic data flow in the script and require designer intervention. The problem is that simulation is still often seen as a post-rationalization tool which is performed separately after the design process is complete. Therefore, the role of the simulation tools should be rethought as process-driven so that they would support dynamic design processes. The development of these process-driven simulations is still in its early stages. (Malkavi, 2005) Nonetheless, integration of AID and simulations can facilitate process-driven simulations.

In addition, personal derives from Case Studies were formulated. First of them is a necessity for agile planning before scripting. During planning, it is recommended to break down the design problem to sub-problems and their functional parts. Planning results should include elaborating parametric and generative schemas.

Besides, graphs should be considered as scalable systems. It implies both scalable bottom-up development type and graph functionality afterwards. In other words, breaking down the design problem on the planning phase should be followed in the graph development in the opposite order. Especially huge graph parts should be elaborated rather separately with its own system inputs and outputs than feeding all-in-one overloaded script. Then these parts can be combined either through custom node creation or group definition.

Other notes from Case Studies analysis contribute to the leaner design process. They include running graphs in manual mode and turning off unengaged parts, regular backup and following naming rules, packaging external tools utilized in the script and providing manual when sharing.

6.3. AID THEORETICAL SYNTHESIS: HYBRID DESIGN PROCESS

In this section, the Conceptual Framework is reviewed based on former research, including Case Studies. Besides, further discussion of aspects which considered to be essential for the development of AIM is conducted. These aspects include AIM integration and development, AID influence, potential issues and new opportunities of AID, and collaboration possibilities within AID.

6.3.1. Conceptual Framework.

The result of the research is concluded in the Conceptual Framework. Proposed Framework in the Introduction chapter was mostly affirmed after the conduct of research with some additional notes. The proposed Framework, AID incapsulates associative part of the digital design with the systematic modelling approach. In addition, after Case Studies analysis it has been realized that AID is primarily the creative process enabling a design logic symbiosis between the human mind and computer for creative investigation and operating on whole design system space in the diversity of possible workflows defined by designer. In contrast, the former CAD methodologies imply computer as an assisting tool presenting static design options only. This concept allows relating AID to Hybrid Design (HD) process. The term refers to the design that utilizes many digital and traditional techniques and design strategies. HD is derived from the term “Hybrid process” suggested by Sevaldson. According to Sevaldson (2005, p. 10): “This new and richer design process ambulates between rationality and intuition, between research and exploration, between the casual and the heuristic, the linear and the networked.“ (Sevaldson, 2005)

Consequently, additive and even traditional crafting design in symbiosis with other design methods should be considered as part of Hybrid Design. Particularly through the lens of AID, algorithm-aided information modelling involves traditional crafting methods on planning stage as it requires parametric and generative schemas to enable design process. Besides, AIM can be developed from the Geometry-Based model. From the other hand, AIM methods can be primarily applied for form-finding and simulations in the preliminary design stage and then be substituted by post-editing with Geometry-Based methods. Geometry-Based Design also has its advantages comparing to AID. GBD is simply more flexible than AID because the designer doesn’t need to choose the type and behaviour of a modelling item. As seen from the above, Hybrid Design approach does not suggest new methodology by itself but rather engages designers to think creatively over methods and tools adopting them under specific needs.

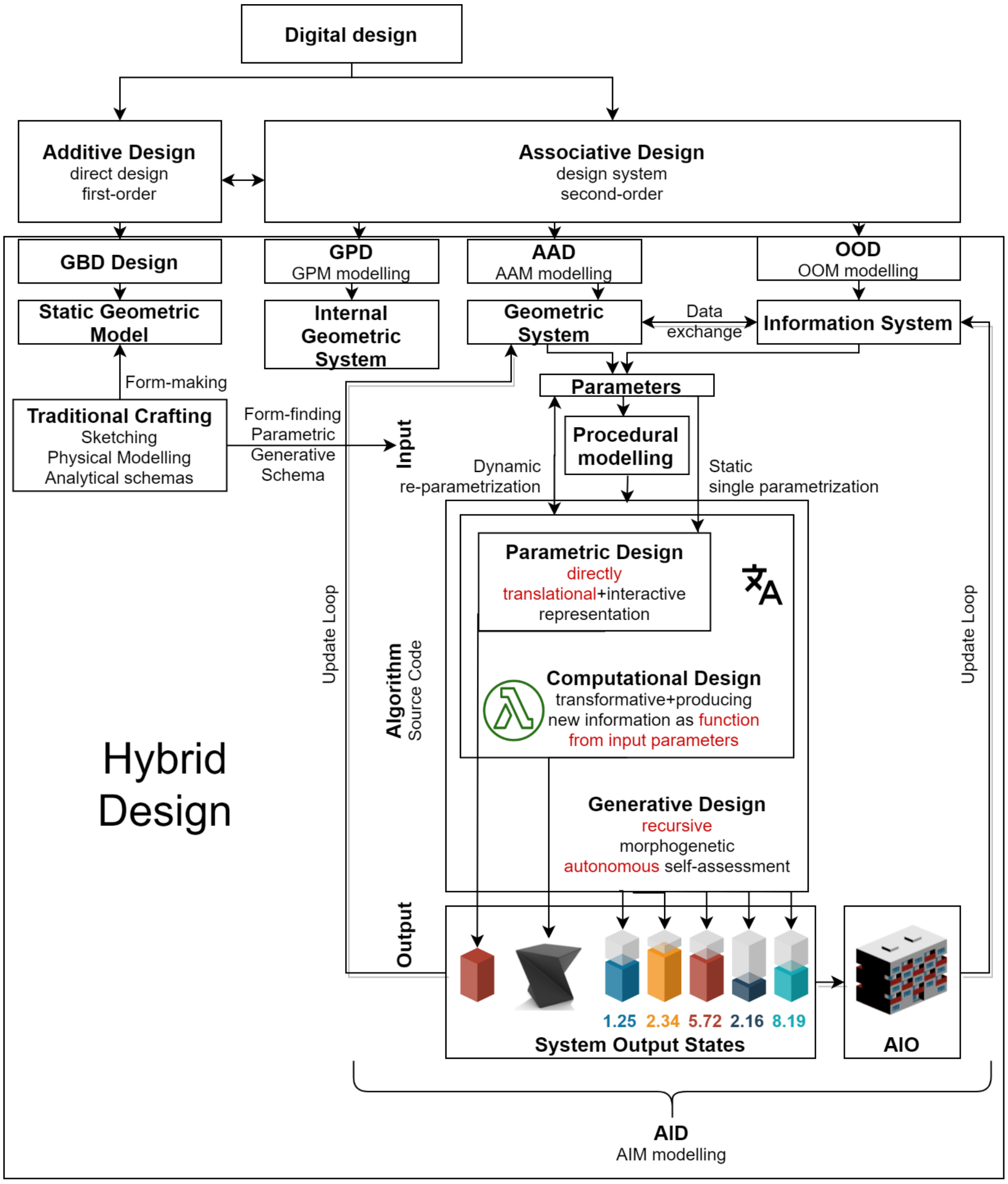


Figure 6.1 – Digital Design Framework

After all, AID should become an essential part of Hybrid Design, encouraging the designer to use tools creatively and systematically at the same time. The revised Conceptual Framework is presented in Figure 6.1.

6.3.2. AIM Integration & Development

In this section, the aspect of AIM integration and its development is considered both in school and industry environments.

SCHOOL.

We are currently undergoing transition times between design methodologies, techniques, and tools in AEC industry. Consequently, architectural pedagogues who are still settling down from the previous tensions between analogue and digital design techniques, in nowadays face another unscheduled cultural shift. While we are moving well into an era of digital design acceptance, we nonetheless operate within a legacy ruled by old-school educators. Currently, there still remains the tension, between the computer as practical “aide-de-camp”, and computer as digital design agent. (Burry, 2011) From my own experience, professors requiring digital representation of projects, still tend to leave the technical part for self-education. One of the reason is insufficient school methodology. I think that digital design should be taught from the experimental point of view and playfully. Therefore innovations in education are mostly coming from inspired young individuals recently graduated.

Subsequently, the transition depends on the new generation of specialists. If we obsess about the need to teach coding skills, we will repeat the mistakes of the 1990s when CAD equalled drafting. Those who want to script need to be taught by computational designers and not by IT specialists. Therefore, the first generation needs to become mature. More crucial still is the need to focus beyond scripting as core feature of AIM methodology to the meta topics: an appropriate approach to learning about the emerging AIM systems in which scripts operate; culturally, and as an emerging theory.

INDUSTRY.

The technical domains have been prominent in the research of architectural practice because, as a useful art, these areas are considered as effecting more immediately the commercial aspects of industry than would design tools, and more conspicuously too. Therefore research budget is rather invested in technical tools development and software, which suggests rapid productivity enhance such as drafting software. The paradox is that for decades digital design software has striven to emulate the analogue working practice that architects developed over the past two centuries. So, architects have not been motivated to shift from analogue design methodologies toward computational methods. As seen from the above, architecture is quite an inert sphere of industry.

However, the last decade has demonstrated the rapid ascendance of more user-friendly design software, easier associative programming languages such as DesignScript. Such alterations caused a growing community of scripters. Nowadays, we can observe a new generation of specialists arising, ready to make their innovative contribution to scripting cultures. (Burry, 2011)

6.3.3. AID Influence

The nature of AID points out the influence in the anticipated fashion. First of all, AID is conspicuously affecting the designer’s toolkit encouraging to experiment. Afterwards, the designer is facing a crisis trying to implement new tools effectively to improve existing workflows. Finally, the need for revising core methodology is emerging. Result of such revising is the present study.

In general, the current trend in the AEC industry is showing a semi-seismic shift from scripting as counter-culture to scripting as a driving force for architectural thinking. (Burry, 2011) Particularly, generative computational methods have led to a shift from form-making to form-finding (Kolarevic, 2005) This means that computer can be utilized to find design solutions within defined constraints of solution space. Generative optimization methods are used in the design, if the solutions space is too large to be explored manually. That’s why the computer is used to explore optimal design on behalf of the designer. (Mueller, 2013) In this fashion, the control has shifted from controlling the design solution to controlling the process itself.

In a nutshell, programmable systems have begun to challenge traditional modes of thinking in architecture and engineering, placing further emphasis on the process ahead of the final result. Just as Darwin had to think beyond form and inquire into the development of biological organisms to understand evolution, so computational methods enable us to rethink how we approach the design process itself. (Harding, 2014)

6.3.4. Potential issues and opportunities of AID

New AID methodology potentially causes a cultural shift. Therefore, from one point of view, Algorithm-aided Information Modelling is considered as next-generation among digital design methodologies, and it provides cogent advantages for AEC industry implementation. From the other point of view, it necessitates to investigate potential issues that may be evoked by this shift. The nature of the problems investigated is mostly inherent from AIM predecessors, BIM and AAM. However, they also include new challenges. Most of the issues are intentionally left as open questions.

THE ROLE OF SCRIPTING IN AID DESIGN PROCESS.

The first issue is related to the role of scripting in the AID design process. There are some much-debated dangers about the standardization of design and creativity lack in digital design due to scripting. One of the main arguments against scripting is the intentional control loss to a computer and the generation of unpredictable results. Indeed, the accidental result emerged without control of the designer is not accepted as a design solution. Creativity is usually seen as an internal character of the designer, so the unpredictable result underestimates the craftsmanship of the designer. However, AID assumes that the designer translates his creativity into design logic investigated upfront. Instead of controlling a static model as a state of his mind at the moment, designer rather controls the concept systematically instructing the design logic to the computer and therefore extending the solution space. Such an approach is seen for many as an opportunity, not an obstacle. It clarifies the idea taking advantages of computation. Any argument that scripting leads to standardisation can be countered by the fact that it allows highly wayward and idiosyncratic designers to innovate in ways otherwise not possible. (Burry, 2011)

Although, always there are extremes, which is better to avoid. Such extremes in AID are laid down within fields of Form-making and Form-Finding. Mr. Laisering defines the place of architecture in digital design between these two extremes:

“Form-making, loosely defined, is a process of inspiration and refinement (form precedes the analysis of programmatic influences and design constraints) versus form-finding as (loosely) a process of discovery and editing (form emerges from analysis). Extreme form-making is not architecture but sculpture […]. Extreme form-finding also is not architecture but applied engineering, where form exclusively determined by function”. (Laiserin, 2008)

The study agrees with this statement. It is vitally important to find the right balance in the architectural design process of AID. Scripting, as one of the primary approaches to AID offers access to whole new ways of exploring design, but the design and creativity always remain at the core.

GLOBAL PARAMETRICISM STYLE QUESTION

Another conspicuous issue is related to the architectural stylistic question. In 2008 Patrick Schumacher declared Parametricism as a successor to Modern architectural Style. Conspicuously Parametricism takes its origin from Parametric Design. Subsequently, it utilizes computational algorithms to manipulate equations for design purposes. One of the main features is that “Parametricism implies that all elements of the design become parametrically variable and mutually adaptive.” According to Schumacher, parametricism is an “Autopoesis”, or a self-referential system, in which all the elements are interlinked and an outside influence that changes one alters all the others." Often Parametric Architecture is associated with flexible shapes and perforations. Although it is the fair characteristic which is caused by frequently differential functions controlling the parametric design systems, it still a one-legged vision of the subject. Parametric design systems can drastically vary both in topology and methods applied.

Alternatively, it is seen from the thesis that parametric models have much more diversity of implementation than the design itself. It can also be applied to DfMA for manufacturing as well as for all range of analysis from building performance analysis to urban analysis which is then used for a design decision. Therefore, even though this study acknowledges Parametricism as architectural style, it emphasizes that the use of AID does not mean that applied methodology will lead to design necessarily framed in Parametrism style. Though, AIM enables the designer to explore solutions that wouldn’t be possible without building parametric system. Hence, AID is particularly able to be used for controlling forms which could not be possible to handle otherwise.

AMBIGUOUS TERMINOLOGY

Next issue is ambiguous terminology due to immature of the methodology and related concepts. Many researchers investigating the same concepts through the different lenses of their subjects apply diversified definitions which is caused by unstructured methodology space so far. However, the situation has become better in recent years but not perfect yet. The field of digital design would benefit from unambiguous terms. Therefore, the taxonomy mapping of the main concepts is conducted in the present study within Conceptual Framework and Glossary in Appendix 1 reflecting the current state of digital design development.

ENFRANCHISING THE AMATEUR PROBLEM.

The last issue considered in this section is the problem of enfranchising the amateur among professionals. Scripted code readily changes hands and, in terms of potential risks, it could become a cloning tool for less talented operators to mimic their masters. In contrast, scripting ought to be the opposite: a liberating design force unleashed by the Internet combining with the innate human desire to share knowledge; from the holistic perspective, the live hive in which the collective critical mass is far greater than the sum of the individuals. Thus scripting causes the rapid development of the field. (Burry, 2011)

6.3.5. Collaboration

The last section of the theoretical part of the research is dedicated to collaboration within AID methodology implemented in the design team. Such a team, except traditional competencies, should include skills such as scripting and digital designing. Further deductions are gathered from the practical part of the thesis as well as from the book Scripting Cultures (Burry, 2011) and based on collaboration types between two mentioned above competencies.

There were three types of collaboration extracted. The first type is via studio teaching where the give-and-take arrangements are quite clear: the computational design master shares knowledge and, in some cases, pieces of the script with protégé who, in turn, works on a project with complexity beyond what he would be likely to address on his own. The studio along with the project benefits from the insight brought to the mix by the project leader's experience. This model is probably the most prevalent, but carries the danger of the flocking towards "familiarity" syndrome – moths drawn to a bright light, or even straightforward design cloning: a shared vocabulary or a sole signature that may not offer individuals sufficient sense of shared authorship. If this is an assertion, it clearly risks coming across as a polemic, given that to some cloning can be read as sharing a creative voice with others. Whether this leads to a risk of a cultural impoverishment through the suppression of individual identities will depend on how critical owning a design is in the age of Web 2.0.

The second collaboration type is working closely with an expert programmer whose background is code writing, not design. This can be fraught with unexpected issues when the computer science-educated code writer has an outlook fundamentally attuned to an analysis of a situation or condition and turns this distilled knowledge into a logical procedure. Unless the designer in the team can, in turn, completely attune their creative synthesis to the proposed logic, the relationship could be one of eternal disconnect with neither party feeling that they have achieved what they set out to do.

The third collaboration type is a variation of the second, where a symbiosis operates between designers and coders. In these teams, the designers are experienced (usually former) scripters who can drill down within their design strategy to a logical basis. They can do this sympathetically as the code writer is sufficiently “design aware” in terms of understanding the implications of creative synthesis not to lock their script into predefined and therefore inevitable outcomes. From experience, this represents a perfect partnership, but they are relatively hard to come by. Put at its simplest, “talented designer seeks clear-headed coder writer” does not mean by itself an automatic success if an intellectual partnership cannot be assured.

7. CONCLUSIONS

In the conclusion of the thesis, Algorithm-aided Information Design as a new holistic design approach is investigated and examined on practice in this study. The AIM methodology aims to synthesize the advantages of well-established BIM and AAM methods and enhances existing Digital Design workflows by the systematic use of Object-Oriented Programming in the AEC industry. Among the core features of Algorithm-aided Information Design are Object-Oriented Programming, custom Shared Mapping Data Exchange, automation of routines, Heuristic and Meta-Heuristic Optimization, generative methods of exploring solution space. All of these features within a single framework allow entitling AID as a new Hybrid methodology which complies with new stage toward Open BIM. The study emphasizes that AID is not necessarily confined to Building Information Modelling but rather refers to the implementation of the computational object-based approach to Digital Design product of any scale. Such a product can be furnishing, mechanical device as well as building, landscape, or GIS system. In this regard, the study includes application of AID methodology to furnishing (refer to Chapter 3), building (refer to Chapter 3,5), and GIS integration (refer to Chapter 2).

AEC Industry has significantly developed over the last decades of the digital age. Architectural design methodologies changed from the digitalizing drawing board to generation of object-based associative systems. However, despite the rapid digital design methods development, traditional techniques still dominate in the construction sector overall world-wide. In the pursuit to deploy digital design achievements into the real construction field, it is crucial to overcome the limitations of existing software ecosystems and specific tools as well as communication methods and way of thinking. The present study is considered as a contribution to this convergence.

Subsequently, the study presents a mixture of exploration through theory review and analysis as well as practical learning and investigation of tools and workflows examination. As a result, the Conceptual Model and Framework are inferred from exploring subject theory through the lens of AID.

Then the practical implementation of AID is examined through experimentation and development of AIM tools and workflows. Consequently, the practical part of the dissertation is divided into four Case Studies, embracing prominent aspects of the AID process and covering Brief Design, Technical Design, and Manufacturing stages. These aspects include model & documentation generation, design automation, performance analysis, coordination, interoperability, manufacturing control, and optimization.

More specifically, in the first Case Study, integration of GIS and OSM data with BIM model using algorithm-aided tools is considered. During the study, Site Model is developed in Dynamo GAE with use of DynaMaps and Elk packages. Final Context Model consists of object entities such as buildings, roads, walkways, and toposurface. Each building instance contains properties imported from OSM, which allow further urban exploration. For example, Site Analysis is conducted shown facility usage within a specified distance from the spot. At the end of Case study, the point cloud is integrated into Site Model for validation and visualization purposes. The point cloud is generated entirely with digital tools originated from Google Earth data, without survey on-site. Photogrammetry approach was applied to achieve the result. Furthermore, the point cloud is exported into 3D Visualization platform – Lumion, for presentation. Though, the discovered method is only suited for preliminary analysis as its accuracy is not sufficient for detailed studies. Generally, the workflow defined in the Case Study allows working with the context within a wide range of scenarios without visiting the site. Such methodology shifts design abilities forward. However, it still does not fully substitute the on-site capturing taking into account the limitation of the data quality available online. Nevertheless, such workflow demonstrates a huge potential as a cheap and fast method for Site Model related design activities. The researcher believes that soon databases will be fulfilled with the necessary datum.

In the second Case Study, documentation authoring is considered. Among the discovered scope of issues solved by AIM, the most demanding are the generation of frequentative elements, sorting and grouping data, design verification, model data management and exchange, generation of reports and quantity take-offs, and graphical views control. As agreed with the partner company, Views Generation and Automatic Layout on a sheet are developed during the study. The workflow includes the development of two tools scripted in Dynamo: CreateViewsByRoom, and LayoutOnSheet. Both tools are equipped with the User Interface providing convenient usage scenarios to the user without necessary scripting expertise. Generally, the second Case Study has shown the advantages of integrating BIM and AAM tools within a single workflow. Meanwhile, important notes are taken into account for future improvements. They are mostly related to view packing problem, including filtering exceeded views and rotation constraint.

Next, the third Case Study examines design-to-fabrication workflow within AIM methodology. One of the outputs is a Panelization tool for converting and transferring manufacturing data from the BIM system to a CNC cutting machine. In this Case Study, the concept of DfMA is introduced. Further, it is applied for the integration of the material-constrain parametric tool into existing BIM model. The elaborated tool allows Panelization of BIM components confined to dimensions of wood panel and layout according to the grain direction. Afterwards, the panelized component layout is exported to CAM software such as Fusion 360 and set for laser cutting. Panelization tool is developed in Dynamo GAE by associative visual programming means as well as by imperative Python scripting means. The UI of the tool is quite simple. It requires to select a component for Panelization, input dimensions of the wood panel, gap values between the panels in layout, and the orientation of panels on sheet. The layout of panels is generated and optimized automatically with sufficient amount of sheets. In this case, the orientation constraint definition is possible as panel geometry can be rotated on the sheet, unlike the former Case Study with views. Once Panelization layout drafting view is exported, the toolpaths are created in Fusion 360, and G-Code is generated for CNC machine instructions. Notably, the developed tool has some limitations, such as cut pattern dependency. The pattern is determined under the specific design needs of the company. Therefore, the tool is proprietary. While it works well if such a pattern is determined, cut pattern integration should be improved for non-proprietary use in the future. In nutshell, prefabrication is introduced but still an ‘alien’ concept in many parts of the AEC industry, as it needs deep investigation and parametric DfMA integration from the early design stages. Continuous use and coordination of digital data models need to be extended all the way from design to digital fabrication and assembly processes. The future development should be focused on DfMA approach and parametric implementation in manufacturing.

Finally, the fourth Case Study concludes all the experience acquired during former Case studies with AIM system development and testing it within the workflow combining different environments and tools. The AIM methodology is examined within the most primer aspects such as procedural modelling, interoperability, object generation, design optimization, and manufacturing. As a result, AIM system is elaborated. It enables accurate and diverse data flows regarding specific needs being controlled from the central model by Graph in Grasshopper. Moreover, the AIM system enables interactive Direct Data Exchange while allows scripting custom Shared Mapping Data Exchange tools. Advantages of both AAM and BIM methodologies are combined within the AIM framework.

In a summary of the practical part of the research, the study shows that AIM as digital design methodology is hard to investigate exceptionally through literature review and exploring theory only. The practical modelling and tools development helped the researcher to systemize knowledge and to achieve a comprehensive vision of the subject overall.

The result of the research is inferred in the Conceptual Framework. Within this framework, AID underlines associative methods of design with extensive use of object-oriented programming as a clue aspect. Besides, after Case Studies analysis, it has been realized that AIM is primarily the creative process enabling a design logic symbiosis between the human mind and computer. It supports the creative investigation and operating on whole design system space in contrast to former CAD methodologies where computer performed as an assisting tool translating static design only. By this way, AID is considered an essential concept within Hybrid Design. In Hybrid Design, tools and methodologies are considered as adaptive. Its approach does not suggest new methodologies by itself but rather engages designers to think creatively over methods and tools, adopting them under specific needs. Meanwhile, the framework provides relationships between methods for the lean design process. After all, AID encourages the designer to use tools and methods creatively and systematically. In this fashion, AID is not a stand-alone methodology but rather can penetrate the existing diversity of Digital Design methods and infer the new qualities on the edge of them.

New AID methodology potentially causes a cultural shift. Therefore, it necessitates research potential issues that may be evoked by this shift. The nature of the problems investigated is mostly inherent from AIM predecessors, BIM and AAM. However, they also include new challenges. Most of the issues are intentionally left as open questions.

One of the problems arisen is related to the role of scripting in the AID design process. The study shows that there are extremes which are better to avoid. Scripting, as one of the primary aspects of AID, offers access to whole new ways of exploring and managing design, but the design and creativity always remain at the core. Therefore, it is vitally important to find the right balance implementing scripting by necessity in the process of AID.

Another issue is due to perceiving and exploitation of parametric modelling. Parametric modelling evoked the emerge of Parametricism style. However, the modelling itself, as part of AID, has much wider implementation beyond the stylistic determination of design. The study emphasizes that the use of AID does not mean that applied methodology will lead to design necessarily framed in Parametrism style. Though, AIM enables the designer to explore solutions that wouldn't be possible without building object-based parametric system.

Next issue is ambiguous terminology due to immature of the methodology and related concepts. The field of digital design would benefit from unambiguous terms. Therefore the taxonomy mapping of the main concepts is conducted in the present study within Conceptual Framework and Glossary in Appendix 1 reflecting the current state of digital design development.

One more issue regarded in the study is the problem of enfranchising the amateur among professionals. Due to the scripting nature of AID, it can produce tools which become the subject of cloning and therefore spreading the immature and arise of authorship issues. From the other point of view, scripting liberates design force unleashed by open source and sharing platforms. In a global sense, it accelerates the development of AEC consolidating gathered knowledge and design solutions among professionals.

The last issue considered in the research is collaboration within AID methodology implemented in the design team. Such a team, except traditional competencies, should include skills such as scripting and digital designing. It can intricate the existing workflows and entangle responsibilities in the team. Subsequently, new roles and interactions should appear. Three collaboration types are defined in this regard. The first is a studio where the leader is a computational design master who shares knowledge with his work team. Another type is a studio where two parties should attune their contribution in order to produce design product: programming expert from one side, and designer from the other side. The third collaboration type is a variation of the second, where a symbiosis operates between designers and coders. In these teams, the designers are experienced scripters who can drill down within their design strategy to a logical basis. From the other hand, coders are "design-aware" and strive to adopt scripts in an agile manner for better tunning of design and solution spaces. From experience, this type represents a perfect partnership, but it is relatively hard to come by.

In a nutshell, Algorithm-aided Information Design is a potential Hybrid approach underlining developing of associative methodologies and integrating it with well-established additive methods of Digital Design. Thus, AID enables creative use of tools and elaborating of agile design systems. The use of object-oriented programming within AID is glue feature which is expected to become the ordinary way of modelling in the near future. Nowadays, AIM is the only modelling method that can control object-based systems through scripting. However, it may engender new reforms in design approaches to come after. The study shows that current AID tools are still quite immature and require much customization. Hence, the extensive engagement and development of tools by vendors as well as by designer community is currently the high-priority task to provide AID lifecycle. Therefore, one of the main intents of the study is a contribution to the popularization of the subject and encouragement for further investigation and development.

8. REFERENCES

Agi, A., 2019. Automatic site modelling. [Online]
Available at: https://www.linkedin.com/feed/update/urn:li:ugcPost:6452527333373935616/

Ahlquist, S. a. A. M., 2011. Computational Design Thinking. s.l.:John Wiley & Sons, Ltd.

Aish, R., 2013. DesignScript: Scalable Tools for Design Computation. Computation and Performance, Volume 2 - Simulation, Prediction and Evaluation(eCAADe 31), pp. 87-95.

Aish, R., 2013. First Build Your Tools. In: B. P. T. Peters, ed. Inside Smartgeometry: Expanding The Architectural Possibilities of Computational Design,. s.l.:John Wiley & Sons Ltd, pp. 36-49.

Andersson, D., 2019. Penlect. [Online]
Available at: https://github.com/Penlect/rectangle-packer

Andreasen, M. K. S. L. T. a. S. K., 1988. Design for Assembly. New York: Springer-Verlag.

Anon., 2019. Level of Development Specification. [Online]
Available at: https://bimforum.org/lod/

Anon., 2020. EarthExplorer. [Online]
Available at: https://earthexplorer.usgs.gov/

Anon., 2020. openstreetmap. [Online]
Available at: www.openstreetmap.org

Anon., 2020. RIBA Plan of Work. [Online]
Available at: https://www.architecture.com/knowledge-and-resources/resources-landing-page/riba-plan-of-work

Architecture, A. f. R. i., n.d. KUKA/PRC. [Online]
Available at: https://www.robotsinarchitecture.org/kukaprc

B.S. Baker, D. B. a. H. K., 1981. A 5/4 algorithm for two-dimensional packing.. In: Journal of Algorithms. s.l.:s.n., pp. 348-368.

Bergin, 2011.

BIMe, 2020. 211in Model Uses Table. [Online]
Available at: https://bimexcellence.org/wp-content/uploads/211in-Model-Uses-Table.pdf

Boeykens, S. &. N., 2009. s.l.:s.n.

Boeykens, S., 2012. Bridging building information modeling and parametric design.. In: G. S. R. Gudnason, ed. eWork and eBusiness in Architecture, Engineering and Construction.. London: Taylor & Francis Group, pp. 453-458.

Burry, M., 2011. Scripting Cultures. Chichester: Wiley.

Daintith, J. W. E., 2008. A Dictionary of Computing. 6 edition ed. s.l.:Oxford University Press.

Dias, R., 2020. Case study proposal, Lisboa: NOZ Arquitectura.

Eastman, C., 2008. BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors. Teicholz, P. Sacks, R. Liston, K. ed. s.l.:John Wiley & Sons.

F.Bucci and M. Mulazzani, L. M. o. e. s., 2006. In: Milano: Electa, pp. 204-208.

F.K.R. Chung, M. G. a. D. J., 1982. On packing two-dimensional bins.. In: SIAM J. Algebraic Discrete Methods. s.l.:s.n., pp. 66-76.

Fasoulaki, E., 2008. Integrated Design: A Generative Multi-Performatiove Design Approach, Master thesis. Cambridge: MIT.

Figueiredo, B., 2019. Introduction to computational modelling in BIM. Porto: BIM A+.

G.S. Hornby, H. L. J. P., 2001. Evolution of generative design systems for modular physical robots. Seoul, IEEE.

Goodman, D., 2020. What is Photogrammetry?. [Online]
Available at: https://www.wisegeek.com/what-is-photogrammetry.htm

Harding, J., 2014. Meta-Parametric Design: Developing a Computational Approach for Early Stage Collaborative Practice. s.l.:ResearchGate.

Humppi, H., 2015. ALGORITHM-AIDED BUILDING INFORMATION MODELING: Connecting Algorithm-Aided Design and Object-Oriented Design. Tampere: Tampere University of Technology, School of Architecture.

ICE, 2020. Designing Buildings Wiki. [Online]
Available at:
https://www.designingbuildings.co.uk/wiki/Design_for_Manufacture_and_Assembly_(DfMA)#DfMA_principles
[Accessed 20 06 2020].

Janssen, P. S. R. C. A. B. S. T. B., 2015. Custom Digital Workflows with UserDefined Data Transformations Via Property Graphs. In: H. S. Gero. J. S., ed. Design Computing and Cognition. Gero. J. S., Hanna. S ed. s.l.:Springer International Publishing, pp. 511-528.

Kolarevic, B., 2005. Towards The Performative In Architecture. In: B. M. A. Kolarevic, ed. Performative architecture: beyond instrumentality. s.l.:Spon Press, pp. 204-214.

Laiserin, J., 2008. Digital Environments for Early Design: Form-Making versus Form-Finding. Cambridge, Harvard University Graduate School of Design, pp. 235-242.

Malkavi, A. M., 2005. Performance Simulation: Research And Tools.. In: B. M. A. Kolarevic, ed. Performative architecture: beyond instrumentality.. s.l.:Spon Press, pp. 85-96.

Mine Özkar, S. K., 2008. Introduction to Shape Grammars. s.l.: SIGGRAPH 2008.

Moreau, S., 2015. Revit model management in Excel. [Online]
Available at: https://www.bim42.com/2015/02/revit-model-management-in-excel/

Mueller, V. S. M., 2013. Generative Components And Smart Geometry. In: B. P. T. Peters, ed. Generative Components And Smart Geometry. s.l.:John Wiley & Sons Ltd., pp. 142-153.

NBS, 2014. s.l.: s.n.

NIERI, L. B. /. G., 2020. KUWAIT INTERNATIONAL AIRPORT. ENGINEERING AND FABRICATION OF A. In: Fabricate. London: UCLPRESS, pp. 84-91.

P.Schumacher, 2010. The Autopoiesis of Architecture, A New Framework for Architecture. pp. vol. I, p. 352.

Quirk, V., 2013. A Brief History of BIM. [Online]
Available at: https://www.archdaily.com/302490/a-brief-history-of-bim

RealityCapture, 2020. RealityCapture tutorial. [Online]
Available at: https://www.youtube.com/channel/UCIfMxiWnFHwmxrm2nNWK2hg

Reilly, C., 2018. Introduction to Associative Design. [Online]
Available at: https://www.lynda.com/Rhino-tutorials/Introduction-associative-design/599608/669559-4.html

Santos, L. a., 2012. s.l.:s.n.

Scheurer, F., 2013. Inside Smartgeometry: Expanding The Architectural Possibilities of Computational Design. In: B. P. T. Peters, ed. Encoding Design. s.l.:John Wiley & Sons Ltd., pp. 186-195.

SeismoSoft, 2020. Civil Engineering Software for Structural Assessment & Structural Retrofitting. [Online]
Available at: https://seismosoft.com/products/seismostruct/
[Accessed 05 04 2020].

Sevaldson, B., 2005. Developing Digital Design Techniques. Investigations on Creative Design Computing. Doctoral thesis.. Oslo: School of Architecture and Design.

Silver, M., 2006. Programming Cultures. New Jersey: Wiley.

Sleator, D., 1980. A 2.5 times optimal algorithm for packing in two dimensions. Information Processing Letters. In: s.l.:s.n., pp. 37-40.

Stasiuk, D., 2018. Design Modeling Terminology. [Online]
Available at: https://provingground.io/2018/06/13/design-modeling-terminology/

Stehling, F. S. a. H., 2020. New Paradigms for Digital. In: Design Transactions. Rethinking Information Modelling. London: UCLPRESS, pp. 42-49.

Steinberg, A., 1997. A strip-packing algorithm with absolute performance bound 2.. In: SIAM Journal on Computing. s.l.:s.n., pp. 401-409.

Stiny, G. a. G. J., 1972. Shape Grammars and the Generative Specification of Painting and Sculpture. Amsterdam, Information Processing 71: Proceedings of IFIP Congress 71.

Tammik, J., 2016. Module 2 [Unwrapped Elevations From Model Line]. [Online] Available at: http://learndynamo.com/mod2/

Tarjan, E. C. J. a. M. G. a. D. J. a. R., 1980. Performance bounds for level-oriented two-dimensional packing algorithms.. In: SIAM Journal on Computing. s.l.:s.n., pp. 808-826.

Tedeschi, A., 2014. AAD_Algorithms-Aided Design. Parametric Strategies using Grasshopper. Brienza (Potenza): Le Penseur.

Vigo, A. L. a. S. M. a. D., 1998. Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem. In: Meta-Heuristics: Advances and Trends in Local Search Paradigms for optimization. Boston: Kluwer Academic Publishers, pp. 125-139.

Vigo, A. L. a. S. M. a. D., 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. In: INFORMS Journal on Computing. s.l.:s.n., pp. 345-357.

Wong, J. B. a. P., 1987. Two dimensional finite bin packing algorithms.. In: Journal of Operational Research Society. s.l.:s.n., pp. 423-429.

LIST OF FIGURES

APPENDICES

APPENDIX 1: GLOSSARY

TERMS

Here by the main therminology is defined through the lens of study subject.

Associative Design/ Programming. Associative Programming is a set of procedures for solving informational-logical problems based on the programmed realization of associative connections between data stored in the memory of digital computers. (https://encyclopedia2.thefreedictionary.com) Associative Design asserts modeller to operate on an instruction set - algorithm or parametric system - by means of Associative Programming. The designer develops an associative set of rules and processes - or "procedural model" – and feeds the system with input data. Then, input data is translated or/and transformed through procedural environment controlled by algorithms, and the system produces geometrical and meta-data output.

AEC. Architecture, Engineering & Construction industry.

AAD/AAM. Algorithm-Aided Design. The term is used to identify the use of specific algorithms-editors or GAEs to assist in the creation, modification, analysis, or optimization of a design. Algorithm-Aided Modelling refers to modeling methods that are defined through scripting. Also known as Parametric Modelling. (Humppi, 2015)

AID/AIM. Algorithm-aided Information Design/ Algorithm-aided Information Modelling. Terms are presented by the study to describe emerging methodology on the edge of Associative Design between OOD and AAD. AID related to Hybrid Design approach underlining developing of associative methodologies and integrating it with well-established additive methods of Digital Design. AID enables creative use of tools and elaborating of agile design systems. The core feature of AID is the use of Object-Oriented Programming (OOP), which allows operating on an object-based procedural design system. Main aspects include model & documentation generation, design automation, coordination, performance analysis, interoperability, manufacturing control, and optimization.

BIM. Building Information Modelling is a process supported by various tools, technologies and contracts involving the generation and management of digital representations of physical and functional characteristics of places. (https://en.wikipedia.org)

CAD/ Digital Design. Computer-Aided Design refers to "the use of computers (or workstations) to aid in the creation, modification, analysis, or optimization of a design." (https://en.wikipedia.org) Also known as Digital Design.

CAAD. Computer-Aided Architectural Design: software programs are the repository of accurate and comprehensive records of buildings and are used by architects and architectural companies. (https://en.wikipedia.org)

Computational System/Design. Computational System is operated according to “a set of procedures consisting of a finite number of rules, which define a succession of operations for the solution of a given problem.” (Ahlquist, 2011) The distinct of Computational System is that it deduces new data reflecting in the System Output States based on input parameters and algorithmic procedures utilizing functional dependencies.

Design Space (DS). Conceptual Model constraining possible design permutations and highlighting the freedom degree to explore alternatives. DS segregates into subsets: material (physical) DS, social DS, digital DS. (Humppi, 2015)

GAE. Graphical Algorithm Editor is a platform which enables graphical scripting. GAE is a subset of Graphical User Interfaces (GUIs). Also known as Visual Programming/Scripting Interface. (Humppi, 2015)

Generative System/Design. Generative System is defined as a procedural system with certain algorithmic autonomy which allows deploying algorithmic transformations and “re-use embedded sub-systems” for specifically incremental morphogenesis. (G.S. Hornby, 2001) Generative Design is an iterative process that allows generating autonomously multiple design options within Solution Space, and, using advanced algorithms, determine the best solution according to defined fitness. (Stasiuk, 2018)

GPD. Geometry-based Parametric Design. Although it does not provide algorithmic methods for modelling, GPD refers to Associative Design because it uses predefined modifiers for model manipulation (e.g. Modifier List in 3D Max). Altering input parameters, we control modifiers which effects on model output. Though, GPD software leaves the ability for the direct mouse-clicking manipulation as well, e.g. 3D Max.

Graph. Refers to a script that is defined graphically in GAE. Also known as: Directed Acy-clic Graph (DAG) visual script and schema. (Harding, 2014)

Heuristic Method (HM). Optimization method often used to optimise a single design objective. Design Problem should be clearly defined and solution space is foreknown. Hence, the nature of heuristic algorithms is predictive and fits only to specific optimization task. Therefore, HM often integrated into plugins meant to solve specific design tasks. HM can be devided into three subtypes: 1st type solves the problem without external analysis involving, straight logic is used; 2nd type solves problem, based on once run analysis; 3rd type utilizes external analysis updating with each iteration.

TPL. Textual Program Language is a scripting process based on text. Known also as conventional imperative language such as Python, C# and Java (Aish, 2013)

Metadata. Refers to "data that provides information about other data. Many distinct types of metadata exist, including descriptive metadata, structural metadata, administrative metadata, reference metadata and statistical metadata."

Metaheuristic Method (MM). Optimization method used for solving problem that is not clearly defined. The design objective is not integrated into MM. Thus, it can adopt to different objectives and constrains. MM utilizes fitness function which is defined as solution performance. Fitness function is used in iterative search for solution space with method such as Evolutionary Algorithms (EA) and Simulated Annealing (SA). Metaheuristic method is deterministic method.

OOD. Object-Oriented Design is the process of planning a system of interacting objects. OOD presents the enterprise as a community of agents, termed objects. An object is an encapsulation of state (data values) and behavior (operations). The behavior of objects is dictated by the rules and principles associated with its object class. (https://www.sciencedirect.com)

OOP. "Object-oriented programming is a programming paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods." (https://en.wikipedia.org)

Parameter (attribute). Terms "parameter" and "attribute" are used interchangeably to describe "the data that is used to control a geometry or an object" (Humppi, 2015) or "Information passed to a subroutine, procedure, or function". (Daintith, 2008) The parameter can be presented by numerical or textual data. Also known as property and variable.

Parametric System/Design. The Parametric System is defined as "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) A parametric system is composed of parameters, translational functions, and their expressed information, including geometrical and metadata. It functions as a predetermined conversion of input parameters to System Output State in only translational and in no transformative way, for procedural representation purposes.

Procedural Modelling. "A Procedural Modelling process is one that uses an explicit instruction set to produce a model outcome." In the context of Procedural Modelling, all the models are the output states of algorithmic systems. (Stasiuk, 2018)

Relational Database. Refers to a digital database based on the relational model of data, as proposed by E. F. Codd in 1970. (https://en.wikipedia.org)

Solution Space. Conceptual Model constraining possible solutions produced by the design process. Usually refers to generation of computational solutions.

SOS. System Output State is the term proposed by the present study referencing to output model controlled by script and defined at certain input conditions of the parametric system.

VPL. Visual Programming Language. Scripting process based on editing graphical algorithms. Known also as visual scripting (Humppi, 2015) or associative scripting/programming (Aish, 2013) and dataflow programming (Harding, 2014)

TOOLS

Here by tools are defined which were used or mentioned during the study.

CAD SOFTWARE

Rhinoceros. Rhinoceros (Rhino, Rhino3D) is CAD modeling software, which can create, edit, analyze, document, render, animate, and translate NURBS curves, surfaces, and solids, point clouds, and polygon meshes. (https://www.rhino3d.com) Rhinoceros is used in processes of CAD, Computer-Aided Manufacturing (CAM), rapid prototyping, 3D printing and reverse engineering in industries including architecture, industrial design (e.g. automotive design, watercraft design), product design (e.g. jewelry design) as well as for multimedia and graphic design. (https://en.wikipedia.org)

BIM SOFTWARE

Autodesk Revit. BIM software which includes functionality for all disciplines related to building design (Architecture, Landscape, MEP, and Structure). The software allows users to design a building and structure and its components in 3D, annotate the model with 2D drafting elements, and access building information from the building model's database. Revit is 4D building information modelling capable with tools to plan and track various stages in the building's lifecycle, from concept to construction and later maintenance and/or demolition. (https://en.wikipedia.org)

Archicad. BIM software, which is developed by GRAPHISOFT. Archicad is directed into architectural design. (http://www.graphisoft.com/archicad/)

Autodesk Navisworks. 3D design review package for Microsoft Windows. Used primarily in construction industries to complement 3D design packages (such as Autodesk Revit, AutoCAD, and MicroStation). Navisworks allows users to open and combine 3D models; navigate around them in real-time and review the model using a set of tools including comments, redlining, viewpoint, and measurements, photorealistic rendering and PDF-like publishing. Supports BIM with clash detection, 4D, 5D time & cost simulations. (https://en.wikipedia.org)

ALGORITHM-AIDED DESIGN TOOLS

Dynamo. A visual programming extension to Autodesk Revit. (http://dynamobim.org). Dynamo is GAE platform, which utilizes data flow graph diagramming, enabling interaction with the BIM model. Therefore Dynamo is considered in the study as an AID tool. Dynamo is based on DesignScript, which has a common notation for both VPL and TPL combining associative and imperative programming. Subsequently, Dynamo enables scalability for "different computational skills...from abstract to domain-specific, including the support for multi-disciplinary design integration."

Grasshopper. GAE running within the Rhinoceros and is fully integrated with its tools. Grasshopper is GAE platform, which utilizes algorithms. Grasshopper's components allow generating and containing 3D geometry, numeric, textual, audio-visual and haptic information. **(**https://en.wikipedia.org**)** Grasshopper supports parametric modelling for architecture, structural engineering, and fabrication, performance analysis, simulations, and generative optimization. Wide range of available add-ons enabling interoperability with BIM lets to entitle Grasshopper as AID tool.

Rhythm. A set of useful nodes to help Revit project maintain a good rhythm with Dynamo. (https://www.dynamopackages.com)

LunchBox for Dynamo. A collection of reusable geometry and data management nodes. The tool includes nodes for surface panelling, geometry, machine learning, Revit data collection, and more. (https://www.dynamopackages.com)

SteamNodes. A set of various nodes to implement a Dynamo workflow.
(https://www.dynamopackages.com)

Archi-lab. A collection of over 50+ custom packages that vastly extend Dynamo's ability to interact with Revit. Nodes contained in archi-lab package vary from basic list operations to advanced Analysis Visualization Framework nodes for Revit. (https://primer.dynamobim.org)

Clockwork for Dynamo. A collection of custom nodes for the Dynamo visual programming environment. It contains many Revit-related nodes, but also lots of nodes for various other purposes such as list management, mathematical operations, string operations, unit conversions, geometric operations (mainly bounding boxes, meshes, planes, points, surfaces, UVs and vectors) and paneling. (https://primer.dynamobim.org)

Refinery ToolKit. A collection of packages to accelerate generative design workflows in Dynamo & Refinery. **(**https://dynamobim.org**)**

AID ADD-ONS

ELK. A set of tools both for Grasshopper and Dynamo GAEs to generate maps and topographical surfaces using open source data from OpenStreetMap.org and USGS. Elk organizes and constructs collections of point and tag data enabling the creation of curves and other Grasshopper/Dynamo geometry.

DynaMaps. The Dynamo package which provides the geometrical representation of the model environment on the preliminary design stage based on OpenStreetMap data. The tool reveals a straightforward workflow with View Extension Interface loading OSM data directly from Dynamo GAE.

Rhino.Inside. An open-source Rhino WIP project which allows Rhino and Grasshopper to run inside other 64-bit Windows applications such as Revit, AutoCAD, etc.

Speckle. An open-source data platform for automation in AEC based on cloud service that liberates AEC from proprietary file formats & closed source software. **(**https://speckle.systems/**)**

GRevit. Grasshopper plugin that allows to define BIM Elements in Grasshopper or SketchUp and translate them directly to Autodesk Revit or AutoCad Architecture. Grevit follows a one way process so your design model remains the geometrical source of truth. (https://www.food4rhino.com/)

Mantishrimp. A Grasshopper/Dynamo set of User Objects that allows moving geometry between Rhino/GH and Revit/Dynamo. (https://www.grasshopper3d.com)

Rhynamo. An open-source plugin for Dynamo that exposes new visual nodes for reading and writing Rhino 3dm files. (https://provingground.io)

Hummingbird. A set of Grasshopper components that facilitate the creation of Revit native geometry. This process exports basic geometric properties and parameter data to CSV text files which are used to create Revit BIM geometry. The tool also supports importing Revit geometry into Rhino, making a bi-directional workflow possible. (https://www.food4rhino.com)

BIM GeomGym IFC. OpenBIM addon for Rhino and Grasshopper enabling IFC model to be generated and exchanged to ArchiCAD, Revit, Bently, Tekla and any other BIM software with IFC capability. (https://www.food4rhino.com)

GIS2BIM. Set of nodes to load 2D and 3D raster and vector GIS-data from various sources into Revit. (https://dynamonodes.com)

Prorubim Dyno. A tool for work with Dynamo GAE, various automatisation and extending Revit possibilities. Dyno provides versatile and simple possibilities for making work with Dynamo workspaces faster. It allows to organize, deploy, and run Dynamo workspaces with custom UI. (http://prorubim.com)