Technische Universität München

TUM School of Engineering and Design

Lehrstuhl für Computergestützte Modellierung und Simulation

Enhancing Prefabricated Building Design with BIM-based Modularization and Automated Transformation: A Case Study on Frame-Tube Structures

Masterthesis

für den Master of Science Studiengang Bauingenieurwesen

Autor: [redacted]

Matrikelnummer: 03750419

1. Prüfer: Prof. Dr.-Ing. André Borrmann

2. Prüfer: Jiabin Wu

Ausgabedatum: 25. April 2023

Abgabedatum: 29. November 2023

Abstract

To enhance the standardization of prefabrication, modular design oriented to Design for Manufacturing and Assembly (DfMA) has been widely adopted in the Architecture, Engineering, and Construction (AEC) industry in recent years by designing integrated volumetric modules instead of individual building components for prefabrication and on-site installation. However, the oversized or overweight modules escalate logistical challenges, which in turn adversely affect design flexibility, cost control, and environmental sustainability. Furthermore, standardization leads to a significant amount of repetitive design work. To narrow these gaps within the context of prefabricated frame-tube structures, a new design approach is proposed through modularization and automated transformations, both of which are based on Building Information Modelling (BIM). In BIM-based modularization, a modularized floorplan is designed to derive different component-level floorplans that enable designers to combine both volumetric modules and individual components. These component-level floor plans are preliminary, while complete component-level floor plans are further generated through automated transformations, for which a design workflow is established. In this workflow, building information is initially automatically extracted from Industrial Foundation Class (IFC) files, further processed, stored in a graph data model, and retrieved using a graph database. After analyzing and integrating the data through established transformation rules, an automated BIM-based transformation scheme is generated. The design of a prefabricated residential building is presented as a case study to verify the BIM-based modularized design process and the automated design workflow, and finally, it implements automated transformations by developing a prototype in the BIM authoring tool. This proposed approach can enhance design standardization and design efficiency for prefabricated frame-tube structures.

Zusammenfassung

Um die Standardisierung der Vorfertigung zu verbessern, hat sich die DfMA-orientierte Modulbauweise in den letzten Jahren in der AEC-Industrie durchgesetzt. Dabei werden integrierte volumetrische Module anstelle einzelner Gebäudekomponenten für die Vorfertigung und die Installation vor Ort entworfen. Die übergroßen oder übergewichtigen Module stellen jedoch eine große logistische Herausforderung dar, was sich wiederum negativ auf die Flexibilität des Designs, die Kostenkontrolle und die ökologische Nachhaltigkeit auswirkt. Darüber hinaus führt die Standardisierung zu einem erheblichen Anteil an sich wiederholender Entwurfsarbeit. Um diese Lücken im Zusammenhang mit vorgefertigten Rohrrahmenkonstruktionen zu schließen, wird ein neuer Entwurfsansatz durch Modularisierung und automatisierte Transformationen vorgeschlagen, die beide auf BIM basieren. Bei der BIM-basierten Modularisierung wird ein modularisierter Grundriss entworfen, um verschiedene Grundrisse auf Komponentenebene abzuleiten, die es den Designern ermöglichen, sowohl volumetrische Module als auch einzelne Komponenten zu kombinieren. Diese Grundrisse auf Komponentenebene sind vorläufig, während vollständige Grundrisse auf Komponentenebene durch automatisierte Transformationen generiert werden, für die ein Planungsworkflow eingerichtet wird. In diesem Workflow werden die Gebäudeinformationen zunächst automatisch aus IFC-Dateien extrahiert, weiterverarbeitet, in einem Graphdatenmodell gespeichert und über eine Graphdatenbank abgerufen. Nach der Analyse und Integration der Daten durch festgelegte Transformationsregeln wird ein automatisiertes BIM-basiertes Transformationsschema erstellt. Der Entwurf eines vorgefertigten Wohngebäudes wird als Fallstudie vorgestellt, um den BIM-basierten modularisierten Entwurfsprozess und den automatisierten Entwurfsablauf zu verifizieren, und schließlich werden automatisierte Transformationen durch die Entwicklung eines Prototyps im BIM-Authoring-Tool implementiert. Der vorgeschlagene Ansatz kann die Standardisierung des Entwurfs und die Effizienz des Entwurfs für vorgefertigte Rohrrahmenkonstruktionen verbessern.

Table of Contents

Acronyms

AEC	Architecture, Engineering, and Construction
API	Application Programming Interface
AI	Artificial Intelligence
BIM	Building Information Modelling
DfMA	Design for Manufacturing and Assembly
GUID	Global Unique Identifier
IDE	Integrated Development Environment
IFC	Industrial Foundation Class
IoT	Internet of Things
MIC	Modular Integrated Construction
PPVC	Prefabricated Prefinished Volumetric Construction
SPF	STEP Physical File

1 Introduction and Motivation

As industrial technology matures, prefabricated buildings have become the core product of construction industrialization. The prefabricated building features mass production of precast building components in the off-site factory for on-site assembly (Araz et al., 2019; Kim et al., 2015). Compared to cast-in-place construction, prefabricated construction offers advantages such as reduced energy consumption, shorter construction periods, and lower labour costs (Jaillon et al., 2014; Li et al., 2014; Qi et al., 2018). Therefore, numerous countries have implemented policies to promote it in facilities with standardized structures (Li et al., 2023). The frame-tube structure is one of the widely used structure types in prefabricated buildings (Zhang et al., 2018; Bian et al., 2022; Li et al., 2023).

1.1 Introduction

In the realm of prefabricated buildings, the design phase significantly influences up to 80% of the operating costs throughout the building's lifecycle (Kovacic et al., 2015). To ensure the above-mentioned benefits of prefabricated buildings, the concept of manufacturing, incorporating requirements for manufacture and assembly into early-stage design, known as DfMA, has been borrowed by the AEC industry (Gao et al., 2020; Lu et al., 2020). However, as a novel design paradigm within the AEC industry, the specification of DfMA has been underwhelming. Currently, due to a lack of knowledge regarding manufacture and assembly, the designers primarily rely on traditional cast-in-situ building design systems, resulting in prefabricated components that are often oversized, overly diverse, and lacking reusability (Huahai et al., 2019). This, in turn, complicates subsequent tasks such as mould creation, fabrication, and installation, ultimately leading to unnecessary additional costs. Thus, a standardized design process plays a pivotal role in prefabricated buildings.

Standardization revolves around the permutation and combination of different generic units with as few specific dimensions as possible, yielding a wide array of adaptable functions and appearances. Within the hierarchy of standardization, modularization stands out at the highest level (Tatum et al., 1987). While the concept of modularization initially gained traction for systems or complex products in mechanical manufacturing, it has since been embraced in the AEC industry to streamline the building design and

construction process (Huahai et al., 2019; Nascimento et al., 2016). One case is the adoption of a new off-site construction technology known as Prefabricated Prefinished Volumetric Construction (PPVC) or Modular Integrated Construction (MIC). It prefabricates volumetric modular systems in the factory instead of previous panel systems for on-site installation (Kamali et al., 2017; Wuni et al., 2019). Each volumetric module therein is an independently operable unit, i.e., a self-contained functional assembly of individual building components designed for use with other such assemblies. However, given project planning, schedules, and costs, the production, transport, and installation of these volumetric modules are limited by their own size, span, and weight (Tak et al., 2020; Hwang et al. et al., 2018; Laovisutthichai et al., 2021). Consequently, despite high standardization, the volumetric modular systems are not the optimal choice for prefabrication. To date, few researchers have explored modularization as a design strategy to promote the standardization of prefabrication. This strategy allows for hybrid prefabrication of the volumetric modules and individual components.

A modern building design cannot be achieved without the support of digitalized technology (Ma & van Ameijde, 2022). As one of the digital tools, BIM provides an integrated approach that incorporates parametric elements of the building's lifecycle (Park, 2011). It is therefore advocated to digitize the modularized design process into BIM. By applying BIM authoring tools, designers can parameterize design modules as spatial placeholders and later replace each of them with prefabricated volumetric modules and prefabricated components with intelligent properties that dynamically respond to changes. The reuse of design modules or combinations of design modules can promote the reuse of prefabricated volumetric modules and prefabricated components. With the predefined parameters and relationships between modules, a standardized prefabricated building model is generated. Moreover, the Industrial Foundation Class (IFC), as a vendor-neutral BIM standard, provides an object-oriented and semantic data model for storing and exchanging building information (Tang et al., 2020). The geometric and semantic information of the prefabricated building extracted from IFC files can support further automation in the design generation.

To further analyze and utilize the modularized building information obtained from IFC files, some form of representation is required. The graph has been acknowledged as an adequate representation of complex data structures (Robinson et al., 2015). One popular form of graph data model is the property graph, which consists of vertices,

edges, and their attributes (Foote, 2022). Based on graphs, the emerging graph databases facilitate the storage and querying of data and their complex internal relationships (Angles, 2012). Specifically, neo4j is one of the main current graph database frameworks based on property graphs (Saad et al., 2023; Van et al., 2023). Data retrieved from graph databases are analyzed and integrated for further BIM-based building design.

Developing construction-centric BIM through manual modelling is both time-consuming and error-prone (Liu et al., 2018). Therefore, secondary development of BIM is used for automated modelling. The repetitive use of prefabricated modules or prefabricated components is accompanied by repetitive design tasks, such as transformations. Plug-ins for BIM can address domain-specific needs (Saad et al., 2022), which in this study are automated transformations after integrating and analyzing graph data to generate complete standardized floor plans for frame-tube structures.

1.2 Thesis Scope

The thesis endeavours to devise a systematic approach to the standardized design of prefabricated frame-tube structures. The primary aim of this thesis is to explore and establish a BIM-based design process through modularization and automated transformation in the context of frame-tube structures.

A key objective is to demonstrate the viability and benefits of incorporating digital technologies, especially BIM, into modularization and transformation. This involves the use of BIM for enhancing the standardizing and efficiency of the design phase, a stage that significantly impacts the operational costs throughout a building's lifecycle. The modularization focuses on the integration of standardized, reusable, and parametric design modules into architectural design to address a critical gap in the current design practices within the AEC industry. Central to automated transformation is the development of an automated design workflow, which leverages the extraction of building information from IFC files, processes this data to create a graph data model, and utilizes graph databases for efficient retrieval and analysis. Finally, the automated transformations can be implemented by developing a plug-in for the BIM authoring tool.

Through this thesis, it is anticipated that a more streamlined, cost-effective, and environmentally sustainable approach to standardized prefabricated building design will be established, setting a precedent for future projects and studies in this rapidly evolving field.

1.3 Thesis Structure

The rest of the paper is organized as follows. Chapter 2 conducts related work on this study. Chapter 3 describes the modularized design process**.** By configuring customized design modules, customized standard space units, customized modularized floorplans, and customized space unit division schema step by step, the component-level floorplans can be initially generated. In addition, based on this design approach, the subsequent design can be automatically completed in the following chapters. Chapter 4 describes the automated design workflow for automated transformations. Firstly, building information regarding modules and building components is extracted from the existing IFC file. Following this, data are further processed to build a property graph. After the graph is generated, the graph-structured data is retrieved with a graph database. To this end, data regarding automated transformations are analyzed and integrated for the implementation of design automation. In Chapter 5, a case for prefabricated residential building design is introduced to implement the design automation based on the design modularization. With the developed prototype in the BIM authoring tool for the automated data transformation, the complete floorplans of this case are automatically generated. Chapter 6 concludes the whole thesis and discusses the future work.

2 Related Work

2.1 Building Modular Design

2.1.1 Modular Buildings

Prefabricated building components are modularized in factories and then transported to the construction site for installation (Kim & Park, 2013). Buildings generated through this modular design approach are referred to as modular buildings.

There are several significant advantages to adopting modular buildings, including fast construction speed, cost-effectiveness, and environmental sustainability. Modular construction can reduce building time by up to 50% compared to traditional methods (Xu et al., 2020). Cost savings are also significant in construction waste and improved resource efficiency (Jaillon & Poon, 2009). Furthermore, modular construction's contribution to sustainability is underscored by its ability to minimize on-site waste and reduce the overall carbon footprint of building projects (Gibb & Isack, 2003).

Despite its advantages, modular construction faces several challenges. One major limitation is the perception of lower quality compared to traditional construction methods (Rausch et al., 2020). Intra-module and inter-module connections play a key role in the overall performance and stability of the structure (He et al., 2023) and are therefore a major challenge in quality control of modular buildings. Additionally, logistical challenges, such as the transportation of modules and the need for specialized equipment for assembly, can pose significant constraints (Tam et al., 2004). Moreover, architectural limitations in terms of design flexibility are highlighted. Modularization can sometimes restrict creative freedom due to standardization (Parker, 2017).

In the future, modular construction will be closely integrated with technologies, such as BIM and advanced manufacturing techniques can enhance precision in the fabrication of modules and offer greater design flexibility (Xiao & Bhola, 2022).

2.1.2 DfMA-Oriented Modular Design

DFMA considers manufacturing early to shorten product development time and ensure a smooth transition to manufacturing, resulting in faster time to market (Bayoumi, 1999).

Products generated by DFMA can be quickly assembled from fewer standardized parts. Parts are designed to be easy to manufacture and common to other designs. The core principles of DfMA-oriented modular design involve optimizing the design for ease of manufacturing and assembly. These principles are applied specifically in modular construction, streamlining the design process (Lu et al., 2021).

DfMA-oriented modular design has found applications in various construction projects. It is used in residential building projects, noting improvements in construction speed and quality (Pan et al., 2012). The DfMA-oriented modular design enhances construction efficiency and worker safety (Lu et al., 2017). It is also applied in sustainable building projects, playing a crucial role in minimizing environmental impact (Jaillon & Poon, 2009). The reduction in construction waste and improved cost-effectiveness are key benefits (Tam et al., 2004).

Despite its advantages, DfMA-oriented modular design faces challenges. The adoption of DfMA in the traditional construction industry faces resistance to change and the need for skilled labor (Kamar et al., 2011). Increased capital costs in the early adoption phase and designers' lack of manufacturing knowledge also limit the use of DfMA in projects (Gao et al., 2020).

Future research should focus on further integrating technology, such as BIM, to enhance the effectiveness of DfMA in modular construction (Dong et al., 2023). DfMA-oriented modular design represents a significant step forward in modern construction methodologies, offering efficiency, sustainability, and cost savings. While challenges remain, ongoing advancements and research continue to broaden its applications and effectiveness.

2.2 Prefabricated Frame-Tube Structure

The concept of prefabricated buildings with frame-tube structures has garnered significant interest in the construction industry due to its efficiency, strength, and sustainability. The frame-tube structure in prefabricated buildings is a composite system combining a rigid frame with a central tube. A comprehensive overview of the structural engineering principles behind this design is provided, emphasizing its ability to resist lateral loads effectively (Taranath, 2010). The optimal design of high-rise buildings uses tube structures (Sarkisian, 2016).

Prefabricated buildings with frame-tube structures offer numerous advantages. The

efficiency and speed of construction are achieved through prefabrication, which has a positive impact on project timelines and cost management (Lawson et al., 2014). Also, prefabricated buildings with frame-tube structures have a positive impact on sustainability, including material efficiency and reduced environmental impact (Chudley & Greeno, 2018).

The integration of advanced technologies like BIM has further enhanced the construction of these structures. BIM optimizes the design and fabrication process of prefabricated frame-core tube structures (Eastman et al., 2011). The application of BIM facilitates collaboration and streamlining workflows in prefabricated construction projects (Darko et al., 2020).

Despite their benefits, prefabricated frame-core tube structures face certain challenges. Due to the complexity of the structural system and the lack of tools, the schematic design of framed tube structures is a difficult task that relies on the experience and domain knowledge of specialists (Fei et al., 2022). Traditionally, senior engineers with sufficient design experience and knowledge are responsible for manually developing the schematic design based on the layout, height, seismic risk, and wind loads, laying the groundwork for subsequent design phases. Additionally, planning and coordination are important to mitigate potential issues (Hamid et al., 2008).

Recent innovations in this field have focused on enhancing structural performance and adaptability. The integration of green building concepts in frame-core tube structures promotes environmental sustainability (Yeang, 2007). The potential of smart technologies and IoT enhances the functionality and adaptability of prefabricated buildings (Casini, 2016). Prefabricated buildings with frame-core tube structures represent a significant advancement in modern construction, offering benefits in terms of efficiency, strength, and sustainability. While challenges exist, ongoing innovations and technological integration continue to enhance their application and performance.

2.3 BIM in Building Design

2.3.1 BIM in Prefabricated Building Design

BIM is a digital representation of a building's physical and functional characteristics. It has gained prominence in recent years for its potential to enhance the efficiency and effectiveness of the design and construction processes in the context of prefabricated building projects.

BIM technology has found extensive applications in various aspects of prefabricated building design. BIM has been utilized to create detailed 3D models of prefabricated components, facilitating accurate assembly and coordination during construction (He et al., 2021). Additionally, BIM has been utilized for clash detection and resolution (Wang et al., 2018), ensuring that prefabricated elements fit seamlessly into the overall structure. The adoption of BIM technology in prefabricated building design offers several significant advantages. Firstly, it enhances collaboration and communication among project stakeholders (Xiao et al., 2022), leading to improved project outcomes. Secondly, BIM enables accurate cost estimation and scheduling, resulting in cost-effectiveness and efficient project management (Chen et al., 2019). Furthermore, it enhances design flexibility and customization of prefabricated elements (Cui et al., 2020).

Despite its benefits, BIM in prefabricated building design faces challenges. One key limitation is the initial investment required for BIM implementation (Sompolgrunk et al., 2023). Additionally, data interoperability issues between different BIM software platforms can hinder seamless collaboration (Aldegeily, 2018). Moreover, the learning curve associated with BIM adoption can be steep for some construction professionals (Ullah & Witt, 2019).

The future of BIM in prefabricated building design is promising, with ongoing technological innovations. Integration with augmented reality (AR) and virtual reality (VR) technologies can provide immersive design and construction experiences (Panya et al., 2023). Furthermore, the use of artificial intelligence (AI) in BIM can enhance decision-making and automation in prefabricated construction (Zhang et al., 2022).

2.3.2 Information Extraction from IFC

IFC is an open and standardized data model used to describe, exchange, and share information typically utilized in the BIM process (Laakso & Kivininiemi, 2012). Various methodologies have been developed to extract information from IFC for different purposes. An approach uses a model server for IFC data, enhancing data retrieval and management (Beetz et al., 2005). Meanwhile, automated information extraction is used for compliance checking, using rule-based methods to parse IFC files (Beach et al., 2015). Recent advancements in IFC data extraction emphasize automation and efficiency. Advanced algorithms for automated data extraction can significantly improve the accuracy and speed of the process (Solihin & Eastman, 2015). Additionally,

machine learning techniques are used to enhance the automation of data extraction from IFC files (Su & An, 2021).

One of the primary challenges in extracting information from IFC is data complexity and volume. The difficulties in handling the extensive and intricate data within IFC files can hinder efficient data processing (Tanyer & Aouad, 2005). Similarly, the challenges in semantic data extraction exist due to the complexity of IFC schemas (Pauwels & Zhang, 2015). In addition, interoperability remains a key issue in the context of IFC data extraction. So far, only a few parsers/tools are available for parsing the most popular IFC STEP physical format (IFC-SPF). At the component level, there are only three free IFC parsers/tools available (Ma & Liu, 2018), including IFC Engine DLL, IfcOpenShell (Mulero-Palencia et al., 2021), and xBIM Toolkit (Zhu et al., 2023). Other higher-level tools are also available, such as BIMServer (Moyano et al., 2021) and IfcWebServer (Pan & Zhang, 2021). The challenges ensure that IFC data can be seamlessly integrated and utilized across different software platforms (Underwood & Isikdag, 2009). The integration of IFC with other data formats can improve interoperability in BIM processes (Nandavar et al., 2018).

2.3.3 BIM Extension and Customization

With technological advancements, secondary development of BIM has become a key means to achieve customized solutions and enhance project efficiency. Common BIM secondary development tools include Revit Application Programming Interface (API) (Jin & Gambatese, 2019) and Dynamo (Divin, 2020). Some auxiliary tools used for Revit secondary development are such as Visual Studio, Revit SDK, Revit, Revit Look up, and AddinManager. The Revit SDK is highlighted as the official software development kit for Revit, containing help documentation for the Revit API and examples with source code (Wang & Hu, 2022). These tools enable developers to create custom plugins and applications to extend the functionality of BIM software. BIM secondary development primarily utilizes technologies such as the .NET framework and the C# programming language (Chen et al., 2016). These technologies support developers in building complex algorithms for data management and automation in design.

There are three applications for BIM secondary development. Firstly, custom tools enable project managers to track progress more effectively, allocate resources, and control costs (Edenhofer et al., 2016). Secondly, secondary development facilitates

design automation, such as automatically generating construction drawings and model analysis, significantly improving design efficiency (Kim,2018). A plugin for quantity calculation is developed based on Revit API.dll, Revit APIUI.dll, and other interfaces for Revit, implementing an external command for quantity calculation through secondary development (Zhang et al., 2022). Thirdly, integrating various data sources and applications ensures seamless information flow and better project collaboration within projects (Pärn & Edwards, 2017). The Revit API is being used to derive the parameter data from the Revit model to carry out calculations for analysis and study on the feasibility of using the Revit API to draft the personnel evacuation program (Fan, 2020).

2.4 Graph in Building Design

2.4.1 Graph, Graph Theory and Graph Database

The integration of graph theory and graph databases in building design and analysis represents a significant advancement in the field. Graphs provide a powerful tool for modelling complex relationships and structures, which is essential in architectural design processes. Graph theory offers a conceptual framework for understanding and designing the spatial structure of architectural designs. A foundational perspective on graph theory is used in analyzing architectural spaces, particularly through space syntax (Hillier & Hanson, 1989). The graph-based analysis is also applied in urban design and the planning process (Turner, 2007). Graph databases have begun to find applications in managing architectural data. Graph databases can enhance BIM effectiveness (Eastman et al., 2011). There is potential for using graph databases to improve data retrieval and management in BIM processes (Pauwels & Terkaj, 2016).

Graphs provide powerful tools for visualization and simulation in architecture. Graph theory underpins much of the computational design and simulation (Kalay, 2004). The collaborative nature of architectural projects benefits significantly from graph theory and databases. The graph databases enhance collaboration among architects, engineers, and builders, particularly in large-scale projects (Janssen, 2006). Graph-based data structures improve interoperability among different software used in architectural projects (Ramonell et al., 2023).

Despite their benefits, the application of graph theory and databases in architecture faces challenges. Implementing graph-based systems in architectural design is

complex (Sengupta, 2011). Future research is likely to focus on integrating advanced computational methods, like machine learning and AI, with graph-based systems in architecture (Kepczynska-Walczak, 2016).

Graph theory and graph databases offer significant potential in transforming architectural design, analysis, and collaboration. Their ability to model complex relationships and structures makes them invaluable in the architectural field. Future advancements in this area are expected to further enhance the efficiency, precision, and creativity in architectural practices.

2.4.2 From IFC to Graph

The transition from IFC to graph-based representations in BIM signifies an advancement in handling, analyzing, and visualizing construction data. Converting IFC data into graph-based models offers numerous advantages in terms of data management and processing. A graph-based approach to IFC data merging has been proposed that emphasizes the complete preservation of relationships between IFC entities in graphical data structures. This approach is essential to maintain the integrity of data relationships during the merging process (Zhao et al., 2020). An interface was developed to convert 2D floor plans into IFC graphic structures. This development enhances the case library of architectural graphics and provides architects with a more comprehensive architectural design and planning tool. The interface bridges the gap between traditional floor plan design and modern IFC graphic structures (Heger & Napps, 2023).

Graph databases can represent IFC data more efficiently, making querying and data processing easier (Khalili & Chua, 2015). Data accessibility and usability are improved in practical applications and methods of this conversion (Zhu et al., 2023). Furthermore, graph-based models significantly enhance BIM functionality, especially in data visualization and interaction. The graph databases facilitate better visualization and interaction with BIM data, resulting in more efficient project management (Gradišar & Dolenc, 2021). Building information models are integrated with monitoring data using graphical data management systems and IFC standard. The focus is on creating an efficient database management system that supports the integration and interoperability of different information models, which is critical for higher-level cyber-physical systems (Gradišar & Dolenc, 2021).

The transition from IFC to graph models is complex. There are technical difficulties in

accurately mapping IFC models to graph structures (Chen, 2019). In addition, ensuring data integrity and consistency during the conversion process is also challenging (Barzegar et al., 2021). Research has been conducted to simplify the complex structure of IFC graphs, which are a combination of various graph types such as tree and network structures. However, it is a challenge to accurately interpret these complex structures to effectively extract and utilize building information (Zhu et al., 2023).

3 Modularized Design Process

The proposed modularized design process is initiated in the early design planning stages, as illustrated in Figure 3.1.

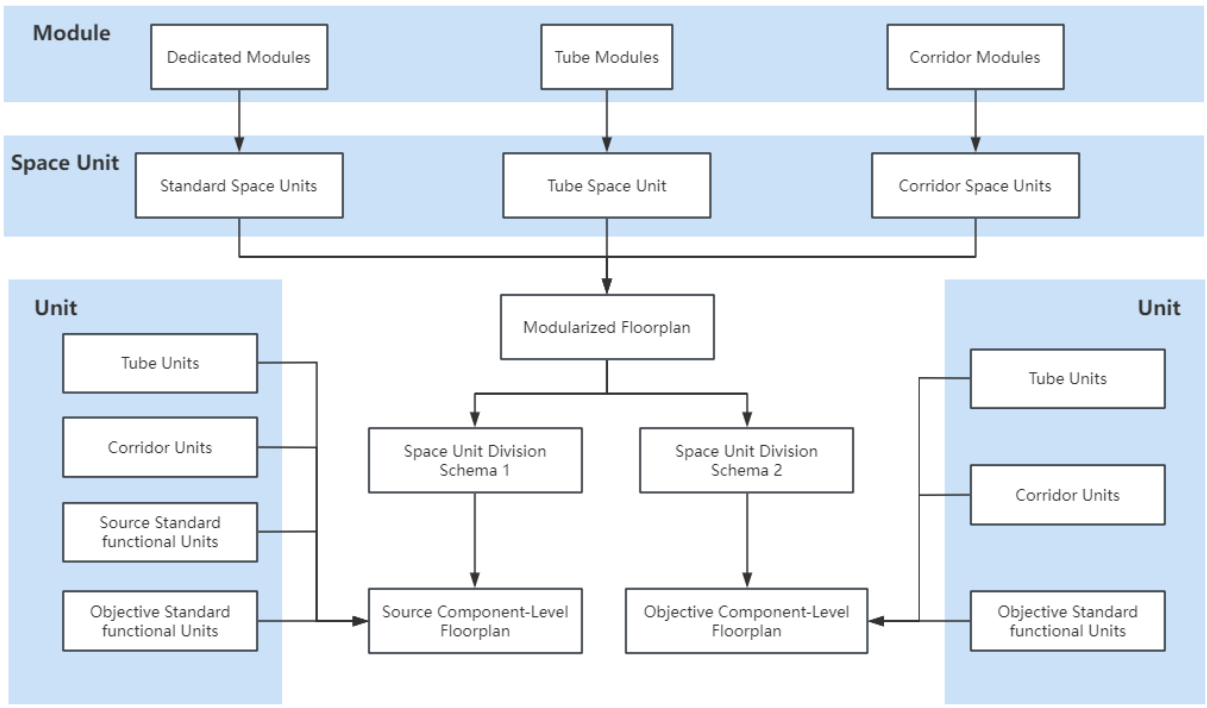

Figure 3.1. Modularized Design Process for Prefabricated Frame-Tube Structures

As illustrated in Figure 3.1, a modularized floorplan is generated in the design process before the generation of component-level floorplans. Design modules are the basic elements for generating the modularized floor plan. Different design modules make up different space units, and then different space units make up the modularized floorplan. For generating a component-level floorplan, the space unit division schemas are formed depending on the modularized floorplan. In conjunction with different building units, the space unit division schema can further generate the preliminary component-level floorplan.

3.1 Design Module

3.1.1 Design Module Definition

In architectural design, design modules function similarly to "Lego" bricks, which are capable of being assembled into complex structures of varied configurations (Cao et al., 2022). These design modules serve as preliminary schematics of the design.

From a technical perspective, design modules are conceptualized as spatial element proxies during the planning phases. Afterwards, each design module is systematically substituted with definitive building components to refine and elaborate the floor plan.

3.1.2 Design Module Feature

Shape: The shape of design modules is uniformly specified. All the design modules are shaped as rectangles in cross-section to form the regular layout of the floor plan.

Dimension: Design modules under the same class can have different dimensions. Designers tailor the sizes of design modules to suit distinctive design projects, taking some considerations such as too large a size restricting the possibilities of spatial configurations and too small a size creating too many joints. With BIM authoring tools, design modules can be parametrized as elements. Each design module is driven by parametric dimensions, including length, width, and height. When changing the value of a parametric dimension, the design module resizes accordingly.

Quantity: Only repeated use of design modules of the same size in one floor plan can contribute to the reuse of prefabricated components. Modularization needs to ensure design variety while increasing the number of design modules of the same size.

Elevation: Design modules are configured on a separate level. Vertically, they are located below all building elements. The elevation of the module is usually the lowest level that houses the building components.

3.1.3 Design Module Classification

The floor plan of a prefabricated building with a frame-core tube structure can be divided into three kinds of units: the tube unit, the corridor unit, and the standard functional units. Correspondingly, in modularization, design modules are organized into three main classes: tube modules, corridor modules, and dedicated modules. The dedicated modules are subclassed into basic modules and conjunctive modules. Design modules of the same class are combined to form specific space units, which are spatial placeholders for specific building units. The three classes of design modules are described as follows:

- Tube module

Tube modules are used to assemble the tube space unit, which is the spatial placeholder for the tube unit of the component-level floor plan. An example of the

configuration process related to tube modules is illustrated in Figure 3.2.

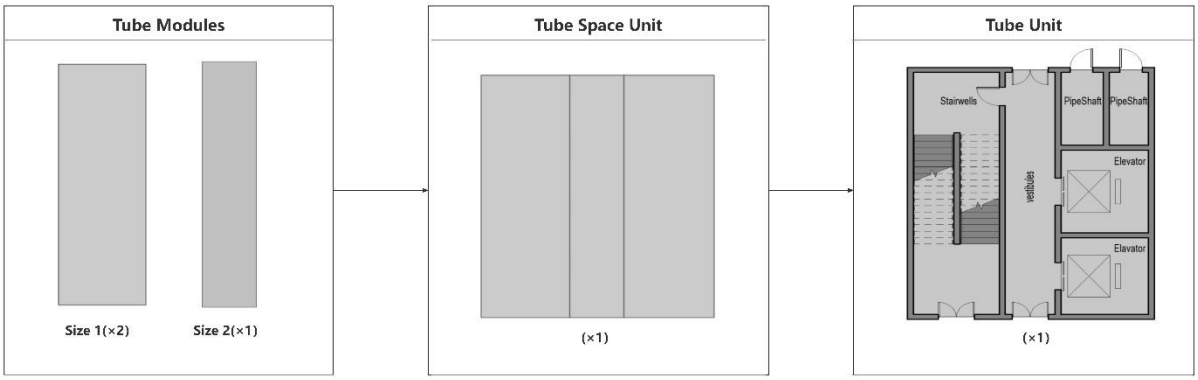

Figure 3.2. Configuration Process related to Tube Modules

- Corridor modules

Corridor modules are used to assemble the corridor space unit, which is the placeholder for the corridor unit of the component-level floor plan. An example of the configuration process related to corridor modules is illustrated in Figure 3.3.

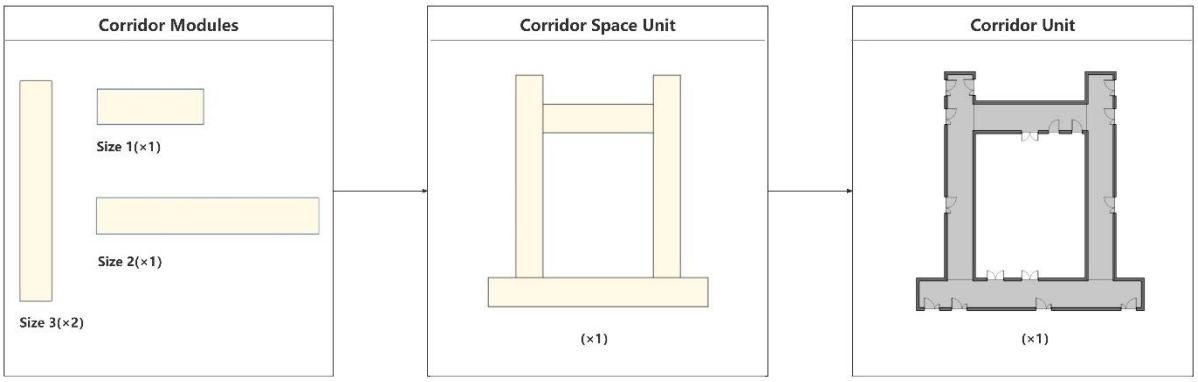

Figure 3.3. Configuration Process related to Corridor Modules

The walls and doors illustrated in the corridor unit of Figure 3.2 are shared parts of the tube unit or standard functional units and, therefore, do not appear repetitively in the complete floor plan.

- Dedicated Modules

Dedicated modules are used to assemble standard space units, which are spatial placeholders for standard functional units of the component-level floor plan. Dedicated modules are the critical design objects in modular design. They are configured for the main objectives of modular design: Firstly, to promote the reuse of prefabricated components, dedicated modules of the same size are reused. Secondly, to increase the multiplicity of standard space units, dedicated modules of varied sizes are combined. Thirdly, to generate an adaptable floor layout, different standard space units

are reused and integrated in a well-organized manner, and further to this, without changing the dimension, quantity or position of modules, standard space units on the field can be flexibly disassembled and reassembled to switch space unit division schemes. To compromise all the above three objectives simultaneously, dedicated modules are subdivided to add additional constraint rules. In modularization, the dedicated modules are further divided into two subclasses: basic modules and conjunctive modules. An example of the configuration process related to dedicated modules is illustrated in Figure 3.4.

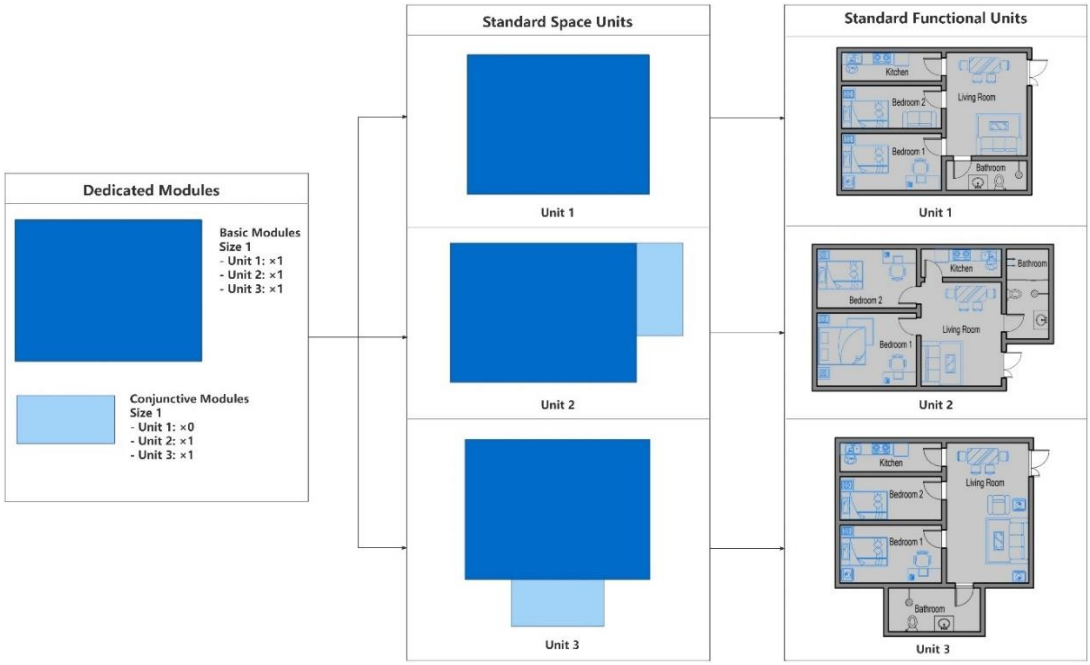

Figure 3.4. Configuration Process related to Dedicated Modules

Basic modules serve as the identifier of each standardized space unit since one standard space unit has one and only one basic module. As illustrated in Figure 3.3, a single basic module can represent a separate standard space unit. Conjunctive modules combined with the basic module to increase the diversity of standard space units. They can be used multiple times in varied sizes within the same unit. They can also have the same dimensions as the basic module. In addition, conjunctive modules play a role in spatial articulation to maintain the coherence and compactness of the design. Of note, conjunctive modules cannot form a unit without the basic module.

3.2 Standard Space Unit

3.2.1 Standard Space Unit Definition

The standard space unit is the spatial placeholder for the standard functional unit of

the component-level floor plan. The standard space unit is non-actual building space.

3.2.2 Standard Space Unit Composition

The standard space unit is a basic module or a combination of dedicated modules. The basic module is unique and irreplaceable within one space unit., while conjunctive modules are auxiliary elements within the same space.

3.2.3 Standard Space Unit Variant

The standard space unit variant is a relative concept. If a standard space unit is known, then the standard space units formed from it by transformations, including translation, rotation and/or mirroring, are all variants of it. A standard space unit and its variants are illustrated in Figure 3.5.

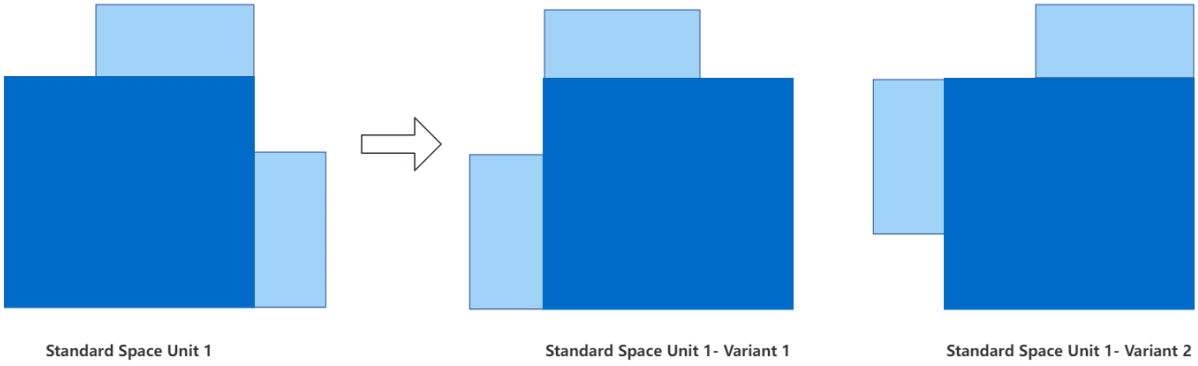

Figure 3.5 A Standard Space Unit and Its Variants

As illustrated in Figure 3.5, variant 1 is obtained by mirroring standard space unit 1, and variant 2 is obtained by rotating standard space unit 1 by 90 degrees counterclockwise.

3.2.4 Standard Space Unit Library

The standard space unit library collects the standard space units and their variants under all space unit division schemes for a specific modularized floor plan. The formation of the module library depends on the designer's preferences and the user's needs.

3.2.5 Standard Space Unit Classification

After the space division schemes are selected in the modularized design, all standard space units are classified into source space units and objective space units. Source space units are the spatial placeholders for the source standard functional units of the

component-level floor plan. Objective pace units are the spatial placeholders for the source standard functional units of the component-level floor plan.

3.3 Modularized Floorplan

The initial task through the modularized design is to generate a modularized floor layout, as illustrated in Figure 3.6.

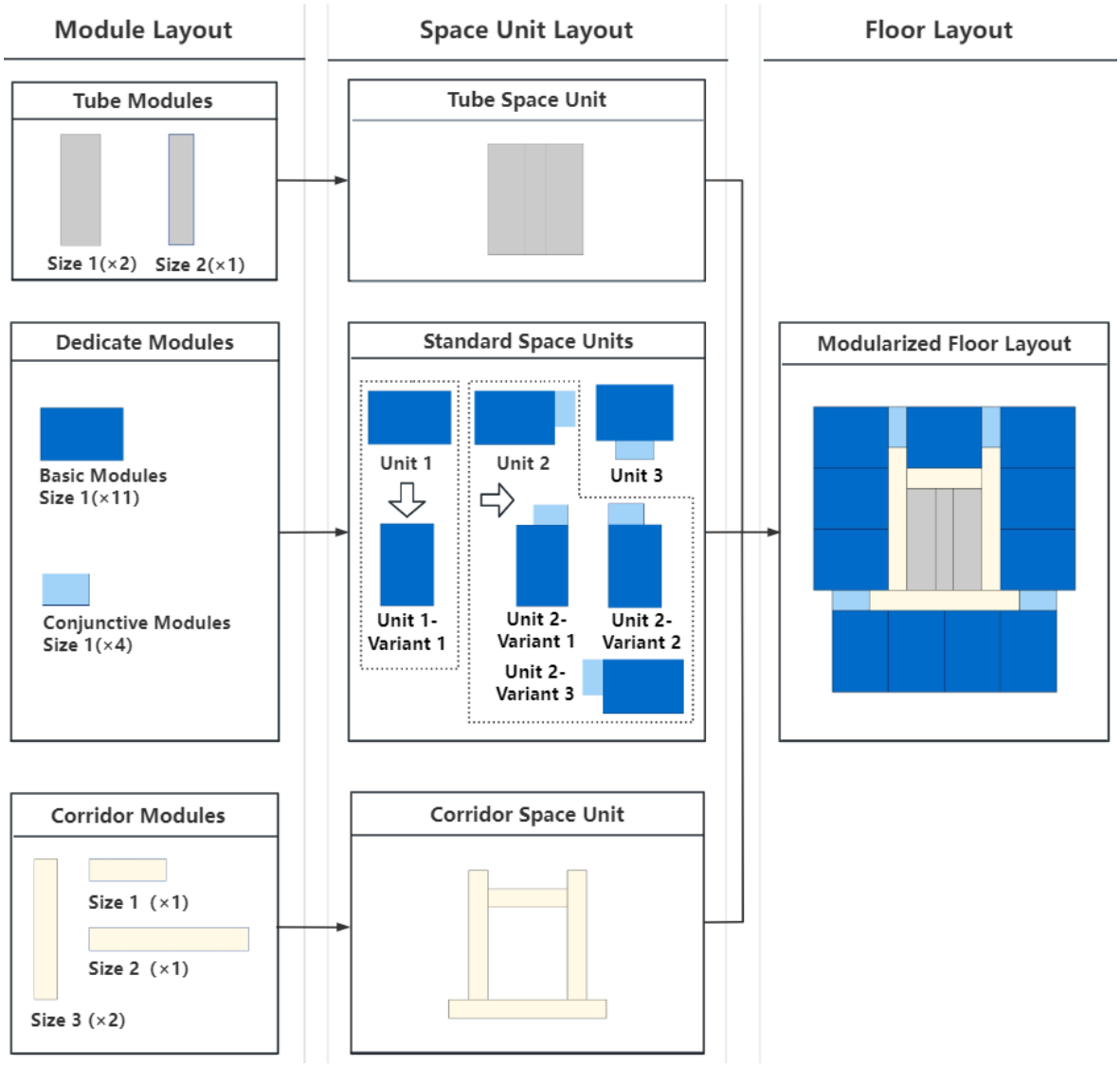

Figure 3.6. Modularized Floor Layout Configuration Process

3.3.1 Modularized Floorplan Configuration

The configuration process of the modularized floor layout includes the following steps:

- Configure the dimension and quantity of design modules by the design module category. All the design modules under the same subclass can be of multiple different dimensions. Design modules under the same subclass can appear multiple times

within a standard space unit except for the basic module. Notably, the basic module is unique within a unit.

- Combine the design modules into standard space units of different shapes, which are placeholders for standard functional spaces of the building. Therein, the variant is the result of mirroring or/and rotating an original standard space unit, e.g., Unit 1-Variant 1 can be the result of rotating Unit 1 by 90° or 180° counterclockwise, and Unit 2-Variant 1 can be the result of rotating Unit 2 by 90° counterclockwise and then mirroring it. Furthermore, since the individual design modules are rectangles, the shape of the standard space units can only be a rectangle, or a polygon composed of rectangles.

- Select the tube space unit, the corridor space unit, and a plurality of standard space units from the standard space unit library. One standard space unit can be used multiple times.

- Combine the selected space units into a modularized floor plan: Locate the tube space unit firstly at or near the centre of the floor plan; Then distribute the standard space units on the outside within the floor plan. Finally, the corridor space unit is sandwiched in between.

3.3.2 Space Unit Division Schema

In a stereotyped floor plan, the dimensions, quantity, and position of design modules are fixed, but the type and quantity of standard space units can vary depending on the different scenarios of module combinations. Based on the example in Figure 3.6, three feasible space unit division schemes are illustrated in Figure 3.7

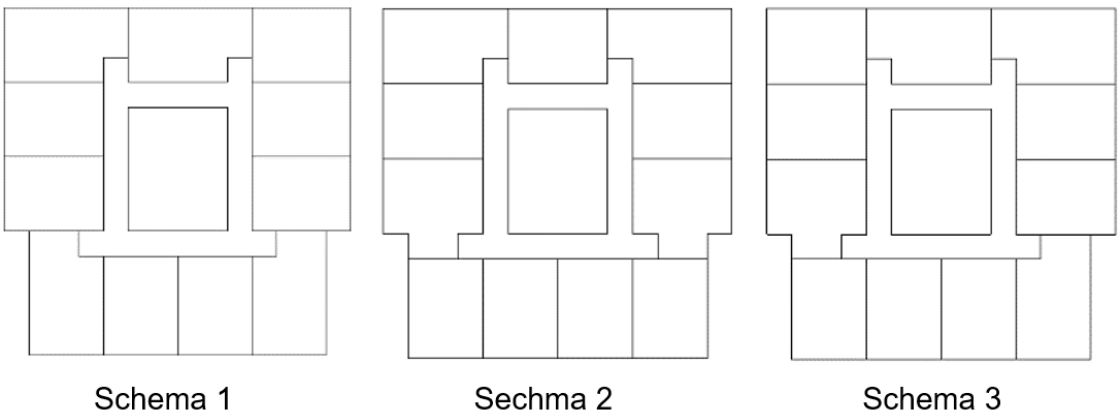

Figure 3.7 Three Space Unit Division Schemes based on Figure 3.6.

The occurrences of standard space units per space unit division schema are illustrated in Table 3-1.

Table 3-1. Occurrences of Different Standard Space Units per Schema

Standard Space Unit	Occurrences of Different Standard Space Units per Schema		
Type	Schema 1	Schema 2	Schema 3
Unit 1	5	3	4
Unit 1 - Variant 1	2	4	2
Unit 2	2	1	0
Unit 2 - Variant 1	1	0	1
Unit 2- Variant 2	1	0	0
Unit 2 -Variant 3	0	1	2
Unit 3	0	2	1

All the standard space units listed in Table 3-1 are from the standard space unit library in Figure 3.6. It can be seen from Table 3-1 that not only the same schemes, but also different schemes share the same standard space units several times. Further, the same standard space units and their variants can be replaced with standard functional units, which have the same internal layout to support prefabrication.

However, multiple occurrences of the same internal layouts within standard functional units also mean a lot of repetitive design work for the designer. To increase the diversity of spatial design, multiple standard space division units' schemes are usually used in the same building project. Considering this, an automated design process is proposed to port the internal layout from one space unit into another by transformations, such as translation, rotation, and mirroring. This transformation can occur within the same schema or from one schema to another.

3.4 Standard Functional Unit

3.4.1 Standard Functional Unit Definition

A standard functional unit is an actual building space with a specific architectural function. For prefabricated buildings, the internal layout of standard functional units has a high degree of repeatability.

3.4.2 Standard Functional Unit Composition

As illustrated in Figure 3.8, the standard functional unit consists of two parts: the

internal layout and the outlined walls. The internal layout of a standard functional unit is composed of different building components, while the outline walls are drawn along the edges of the corresponding standard space unit.

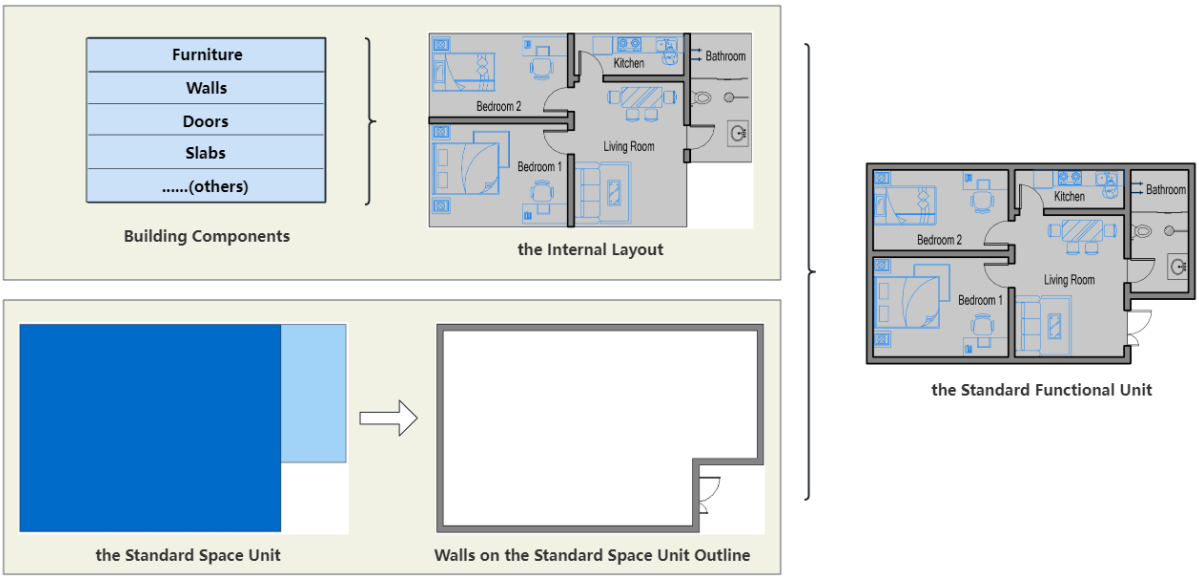

Figure 3.8. The Configuration of One Standard Functional Space Unit

3.4.3 Standard Functional Unit Classification

All standard functional units are classified into source standard functional units and objective functional units. The source standard functional unit is the unit whose internal layout is finished by the designer in the preliminary component-level floor plan. It is located on the source floor. Its internal layout is the object of automated transformation. The objective standard functional unit is the unit whose internal layout is empty in the preliminary component-level floor plans. It is located on both the source floor and the objective floor. It is the objective of automated transformation.

3.5 Preliminary Component-Level Foorplan

Preliminary component-level floorplans are the results of designing through modularization and the prerequisites for automated transformations. The preliminary component-level floorplans contain all the elements to be transformed and the necessary elements related to the transformation.

3.5.1 Component-Level Floorplan Classification

Component-level floorplans are classified into the source floor and the objective floor. Notably, objective standard functional units can exist on the source floor, but source standard functional units cannot exist on the objective floor.

3.5.2 Component-Level Design Rule

Based on the modularized floor layout, the designer can conduct a component-level design on the upper floors, where general design rules for the component-level floorplan compatible with the modularized design layout are proposed as follows:

- The centerlines of all exterior walls on the upper levels are arranged along the edges of rectangular design modules on the lowest level.

- The centerlines or extensions of partial interior walls on the upper levels are arranged along the edges of rectangular design modules on the lowest level.

- The individual prefabricated floor slabs on the upper levels cover the internal area held by design modules on the lowest level.

3.5.3 Preliminary Component-Level Floorplan Configuration

In the preliminary floor plans, the following building components are configured:

- Building components for all source standard functional units, including contour walls and internal layouts

- Contour walls for all objective standard functional units

- Building components for the tube unit and corridor units, including contour walls and internal layouts

By accessing the internal layout of the source standard functional unit, the internal layout of the matching objective standard functional unit can be automatically filled.

4 Automated Design Workflow

The main task of design automation refers to the automated transformation of the internal layout of the source standard functional unit into the matching objective standard functional units. To realize this task, the following data need to be automatically obtained:

- Identification data matching the source standard functional unit and the target standard functional unit.

- The building components within each source standard functional unit.

- The specific transformation type, such as translation, rotation, and mirroring.

- The paired reference points for both the source standard functional unit and the objective standard functional unit

In this chapter, the key information related to transformations is obtained through the design workflow. Based on the modularized design process, the design workflow provides a theoretical basis for the implementation of automated design of prefabricated buildings featuring frame-core tube structures.

4.1 Data Extraction from IFC

The first step in the design workflow is to extract the required component information from the IFC file regarding modules, slabs, furniture, assembled rooms, walls and doors.

4.1.1 Storey Data Extraction

Storeys are under the entity “IfcStorey”. In the IFC hierarchy, the entity inheritance is:

- IfcRoot-IfcObjectDefinition-IfcObject-IfcProduct-IfcSpatialElement-IfcSpatialStructureElement-IfcStorey

The storey elevation is extracted from the IFC file by filtering another attribute, “Name” of IfcStorey, where the actual name of the storey is user-specified.

4.1.2 Module Data Extraction

Modules are custom non-building three-dimensional components under the entity

“IfcBuildingElementProxy”. In the IFC hierarchy, the entity inheritance is:

- IfcRoot-IfcObjectDefinition-IfcObject-IfcProduct-IfcElement-IfcBuildingElement-IfcBuildingElementProxy

Three attributes of module are extracted from the IFC File: IFC_GUID $global\_id\_m$, center coordinates (x_m, y_m), and dimensions (l_m, w_m), as illustrated in Figure 5.1.

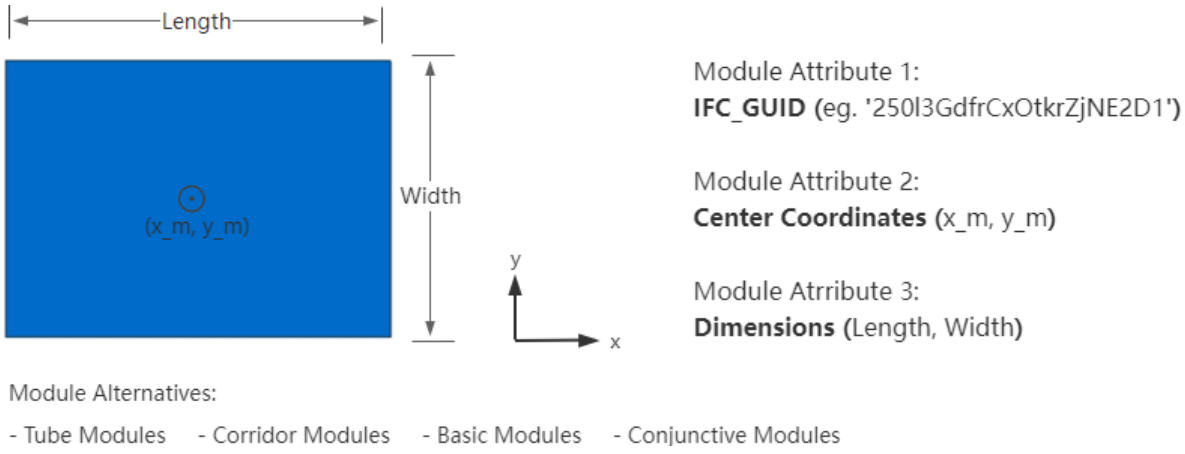

Figure 5.1. Module Information Extracted from IFC File

IFC_GUID is the unique ID of each module, which acts as an identifier. Center coordinates represent the central point of a module's rectangular cross-section on the floor plane, while dimensions refer to its length and width. Coordinates and dimensions are invoked together as vital parameters for calculating connectivity and accessibility between building spaces carried above modules, as well as for capturing building components within the building spaces carried above modules. A partial code to access the three types of data for one module is illustrated in Figure 6.7.

```
01. global_id_m = module.GlobalId
02. x_m = module.0bjectPlacement.RelativePlacement.Location.Coordinates[0]
03. y_m = module.0bjectPlacement.RelativePlacement.Location.Coordinates[1]
04. l_m = module.IsDefinedBy[2].RelatingPropertyDefinition.HasProperties[0].NominalValue.wrappedValue
05. w_m = module.IsDefinedBy[2].RelatingPropertyDefinition.HasProperties[1].NominalValue.wrappedValue
```

Figure 5.2. Partial Code for Data Extraction of One Module from IFC 2×3 Coordination View 2.0

4.1.3 Slab Data Extraction

Slabs are under the IFC entity “IfcSlab”. In the IFC hierarchy, the entity inheritance is:

- IfcRoot-IfcObjectDefinition-IfcObject-IfcProduct-IfcElement-IfcBuildingElement-IfcSlab

Three types of slab data are extracted from the IFC file: IFC_GUID $global\_id\_s$, the elevation e_s, and center coordinates (x_s, y_s). A partial code to access the data of one slab is illustrated in Figure 5.3.

```
global_id_s = slab.GlobalId
e_s = slab.ObjectPlacement.PlacementRelTo.RelativePlacement.Location.Coordinates[2]
x_s = round(slab.Representation.Representations[0].Items[0].Position.Location.Coordinates[0])
y_s = round(slab.Representation.Representations[0].Items[0].Position.Location.Coordinates[1])
```

Figure 5.3. Partial Code for Data Extraction of One Slab from IFC 2×3 Coordination View 2.0

4.1.4 Furniture Data Extraction

Furniture is under the IFC entity “IfcFurnishingElement”. In the IFC hierarchy, the entity inheritance is:

- IfcRoot-IfcObjectDefinition-IfcObject-IfcProduct-IfcElement-IfcFurnishingElement

Three types of furniture data are extracted from the IFC file: IFC_GUID $global\_id\_f$, the elevation e_f, and the specific coordinates (x_f, y_f) placing each furnishing element. A partial code to access the data of one furnishing element is illustrated in Figure 5.4.

```
global_id_f = furniture.GlobalId
e_f = furniture.ObjectPlacement.PlacementRelTo.RelativePlacement.Location.Coordinates[2]
x_f = round(furniture.Representation.Representations[0].Items[0].Position.Location.Coordinates[0])
y_f = round(furniture.Representation.Representations[0].Items[0].Position.Location.Coordinates[1])
```

Figure 4.4. Partial Code for Data Extraction of One Furnishing from IFC 2×3 Coordination View 2.0

4.1.5 Assembled Room Data Extraction

The assembled rooms are prefabricated volumetric modules in the prefabricated building. All components inside the room are prefabricated and assembled into a standardized whole in the factory and then transported to the construction site. Rooms like bathrooms and kitchens are usually designed as assembled rooms. In the same way as modules, assembly rooms are under the entity “IfcBuildingElementProxy”.

Three types of assembled room data are extracted from the IFC File is: IFC_GUID $global\_id\_r$, the elevation e_r, and the center coordinates (x_r, y_r). A partial code to access the data of one assembled room is illustrated in Figure 4.5.

```
global_id_r = room.GlobalId
e_r = room.ObjectPlacement.PlacementRelTo.RelativePlacement.Location.Coordinates[2]
x_r = round(room.Representation.Representations[0].Items[0].Position.Location.Coordinates[0])
y_r = round(room.Representation.Representations[0].Items[0].Position.Location.Coordinates[1])
```

Figure 4.5. Partial Code for Data Extraction of One Assembled Room from IFC 2×3 Coordination View 2.0

4.1.6 Wall Data Extraction

Walls are under the IFC entity “IfcWall”. In the IFC hierarchy, the entity inheritance is:

- IfcRoot-IfcObjectDefinition-IfcObject-IfcProduct-IfcElement-IfcWall

4 types of wall data are extracted from the IFC is: IFC_GUID $global\_id\_r$, the elevation e_w, length l_w, direction ratio r_w and coordinates (x_w, y_w) of the starting point for drawing each wall. The direction ratio varies in different ways of drawing the wall. Notably, the direction ratio r_w for walls drawn from left to right is not directly available. A partial code to access the data of one wall is illustrated in Figure 4.6.

```
01. global_id_w = wall.GlobalId
02. e_w= wall.0bjectPlacement.PlacementRelTo.RelativePlacement.Location.Coordinates[2]
03. x_w = wall.0bjectPlacement.RelativePlacement.Location.Coordinates[0]
04. y_w = wa11.0bjectPlacement.RelativePlacement.Location.Coordinates[1]
05. l_w = wall.Representation.Representations[1].Items[0].SweptArea.XDim
06. r_w = wall.wall.ObjectPlacement.RelativePlacement.RefDirection.DirectionRatios
```

Figure 4.6. Partial Code for Data Extraction of One Wall from IFC 2×3 Coordination View 2.0

4.1.7 Door Data Extraction

Doors are under the IFC entities “IfcOpeningElement” and “IfcDoor”. The inheritance of the two entities are as follows:

- IfcRoot-IfcObjectDefinition-IfcObject-IfcProduct-IfcElement-IfcBuildingElement-IfcDoor
- IfcRoot-IfcObjectDefinition-IfcObject-IfcProduct-IfcElement-IfcFeatureElement-IfcFeatureElementSubtraction-IfcOpeningElement

Notably, since doors are attached to walls, doors can be accessed as an attribute of walls. 2 types of door data are extracted from the IFC is: family type t_{op} and distance d_{op} to the starting coordinate of the wall. A partial code to access the data of one door attached to the wall is illustrated in Figure 4.7.

```
01. op = wall.HasOpenings[0]
02. d_op= op.RelatedOpeningElement.0bjectPlacement.RelativePlacement.Location.Coordinates[0]
03. t_op= op.RelatedOpeningElement.Name.split(":")[1]
```

Figure 4.7. Partial Code for Data Extraction of One Door attached to the Wall from IFC 2×3 Coordination View 2.0

4.2 Data Processing for Graph Data Model

Data processing is a crucial step in the transition from building information to graph data. It involves:

- Establish relationships between modules, including connectivity and accessibility.
- Integrate building components: slabs, furniture, assembled rooms, and walls into each dedicated module within the source standard functional space.
- Integrate building components: walls into the accessibility between two dedicated

modules.

With the extracted and processed data, the final graph data model can meet the requirements for further data analysis.

4.2.1 Building Components Attributed to the Module

Slabs, furniture, and assembled rooms on the source floor are subdivided and assigned to the corresponding dedicated modules by following two steps. Firstly, the components regarding slabs, furniture, and assembled rooms are firstly filtered by the source floor elevation. After that, they are further filtered by following filter conditions, where the subscript c represents "component" and m represents "module".

$$\begin{cases} x_c \in \left(x_m - \frac{l_m}{2},\ x_m + \frac{l_m}{2}\right) \\ y_c \in \left(y_m - \frac{w_m}{2},\ y_m + \frac{w_m}{2}\right) \end{cases} \tag{1}$$

Walls on the source floor are subdivided and assigned to the corresponding dedicated modules by following three steps. Firstly, the walls are firstly filtered by the source floor elevation. Then, 4 types of walls are filtered out depending on different direction ratios: horizontal walls drawn from left to right ("left drawn walls"), horizontal walls drawn from right to left ("right drawn walls"), vertical walls drawn from the bottom to the top ("up drawn walls"), and vertical walls drawn from the top to the bottom ("down drawn walls"). After that, different types of walls are further filtered by different conditions as follows, where the subscript w represents "wall" and m represents "module". And notably, the maximum and minimum wall widths w_{w_min} and w_{w_max} are user-input data.

- "down drawn wall":

$$\begin{cases} x_w \in (x_m - \frac{l_m}{2} + \frac{w_{w\_min}}{2},\ x_m + \frac{l_m}{2} - \frac{w_{w\_min}}{2}) \\ y_w \in (y_m - \frac{w_m}{2} - \frac{w_{w\_min}}{2},\ y_m + \frac{w_m}{2} + \frac{w_{w\_min}}{2}] \\ y_w - l_w \in [y_m - \frac{w_m}{2} - \frac{w_{w\_min}}{2},\ y_m + \frac{w_m}{2} + \frac{w_{w\_min}}{2}) \end{cases} \tag{2}$$

- "up drawn wall":

$$\begin{cases} x_w \in (x_m - \frac{l_m}{2} + \frac{w_{w\_min}}{2},\ x_m + \frac{l_m}{2} - \frac{w_{w\_min}}{2}) \\ y_w \in [y_m - \frac{w_m}{2} - \frac{w_{w\_min}}{2},\ y_m + \frac{w_m}{2} + \frac{w_{w\_min}}{2}) \\ y_w + l_w \in (y_m - \frac{w_m}{2} - \frac{w_{w\_min}}{2},\ y_m + \frac{w_m}{2} + \frac{w_{w\_min}}{2}] \end{cases} \tag{3}$$

- “left drawn wall”:

$$\begin{cases} x_w \in (x_m - \frac{l_m}{2} - \frac{w_{w\_min}}{2},\ x_m + \frac{l_m}{2} + \frac{w_{w\_min}}{2}] \\ y_w \in (y_m - \frac{w_m}{2} + \frac{w_{w\_min}}{2},\ y_m + \frac{w_m}{2} - \frac{w_{w\_min}}{2}) \\ x_w - l_w \in [x_m - \frac{l_m}{2} - \frac{w_{w\_min}}{2},\ x_m + \frac{l_m}{2} + \frac{w_{w\_min}}{2}) \end{cases} \quad (4)$$

- “right drawn wall”:

$$\begin{cases} x_w \in [x_m - \frac{l_m}{2} - \frac{w_{w\_min}}{2},\ x_m + \frac{l_m}{2} + \frac{w_{w\_min}}{2}) \\ y_w \in (y_m - \frac{w_m}{2} + \frac{w_{w\_min}}{2},\ y_m + \frac{w_m}{2} - \frac{w_{w\_min}}{2}) \\ x_w + l_w \in (x_m - \frac{l_m}{2} - \frac{w_{w\_min}}{2},\ x_m + \frac{l_m}{2} + \frac{w_{w\_min}}{2}] \end{cases} \quad (5)$$

4.2.2 Connectivity between Modules

The connectivity between modules is the prerequisite for the connectivity and accessibility between and within building units. Regarding 4 different types of modules, the connectivity between modules has 10 kinds:

- The connectivity between tube modules
- The connectivity between the tube module and the corridor module
- The connectivity between the tube module and the basic module
- The connectivity between the tube module and the conjunctive module
- The connectivity between corridor modules
- The connectivity between the corridor module and the basic module
- The connectivity between the corridor module and the conjunctive module
- The connectivity between basic modules
- The connectivity between the basic module and the conjunctive module
- The connectivity between conjunctive modules

The connectivity information between two modules is represented in Figure 4.7, where module 1 has center coordinates (x_1, y_1), width w_1 and length l_1 and module 2 has center coordinates (x_2, y_2), width w_2 and length l_2. Depending on the different relative positions of modules, the connectivity can be determined as follows:

- If the relative position is “Down&Up” or “Up&Down”, module1 and module 2 that satisfy the following two conditions are connected:

$$\begin{cases} |x_2 - x_1| - \frac{l_1+l_2}{2} < 0 \\ |y_2 - y_1| - \frac{w_1+w_2}{2} = 0 \end{cases} \tag{6}$$

- If the relative position is “Left&Right” or “Right&Left”, module 1 and module 2 that satisfy the following two conditions are connected:

$$\begin{cases} |x_2 - x_1| - \frac{l_1+l_2}{2} = 0 \\ |y_2 - y_1| - \frac{w_1+w_2}{2} < 0 \end{cases} \tag{7}$$

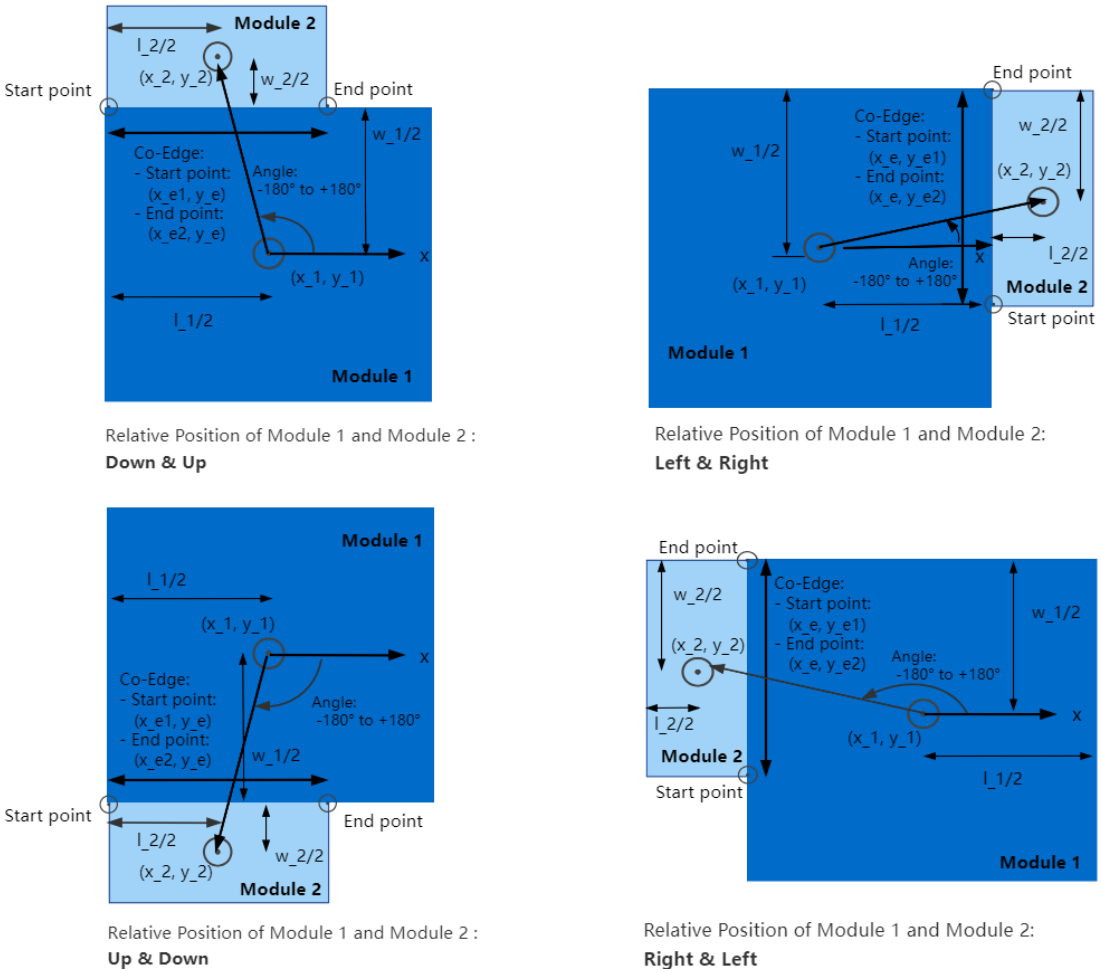

Figure 4.7. Connectivity between Module 1and Module 2 under the 4 Kinds of Relative Positions

As illustrated in Figure 4.7, the connectivity between module 1 and module 2 has 2 attributes: the angle α and the co-edge with a start point (x_{e1}, y_{e1}) and an endpoint (x_{e2}, y_{e2}). The angle α is a directed angle formed by counterclockwise rotation from the positive direction of x-axis to the directed line segment starting at the centre point of module 1 and ending at the centre point of module 2. Generally, $\alpha \in (-180, +180]$.

The co-edge under different relative positions can be calculated as follows:

- If the relative position between module 1 and module 2 are “Down&Up” or “Up&Down”:

  x_{e1}, x_{e2} are selected from 4 values: $x_1 - 1/2\, l_1$, $x_1 + 1/2\, l_1$, $x_2 - 1/2\, l_2$, $x_2 + 1/2\, l_2$, where x_{e1} is equal to the second one of the four values sorted in increasing order and x_{e2} is equal to the third one of the four values sorted in increasing order.

  y_{e1}, y_{e2} are selected from 4 values: $y_1 - 1/2\, w_1$, $y_1 + 1/2\, w_1$, $y_2 - 1/2\, w_2$, $y_{m2} + 1/2\, w_2$, where y_{e1}, y_{e2} are both equal to the second one or the third one of the four values sorted in increasing order. Specifically, for “Down&Up”: $y_{e1} = y_{e2} = y_{m1} + 1/2\, w_{m1} = y_{m2} - 1/2\, w_{m2}$, and for “Up&Down”: $y_{e1} = y_{e2} = y_1 - 1/2\, w_1 = y_2 + 1/2\, w_2$.

- If the relative position between module 1 and module 2 are “Left&Right” or “Right&Left”:

  x_{e1}, x_{e2} are selected from 4 values: $x_1 - 1/2\, l_1$, $x_1 + 1/2\, l_1$, $x_2 - 1/2\, l_2$, $x_2 + 1/2\, l_2$, where x_{e1}, x_{e2} are both equal to the second one or the third one of the four values sorted in increasing order. Specifically, for “Left&Right”: $x_{e1} = x_{e2} = x_1 + 1/2\, l_1 = x_2 - 1/2\, l_2$, and for “Right&Left”: $x_{e1} = x_{e2} = x_1 - 1/2\, l_1 = x_2 + 1/2\, l_2$.

  y_{e1}, y_{e2} are selected from 4 values: $y_1 - 1/2\, w_1$, $y_1 + 1/2\, w_1$, $y_2 - 1/2\, w_2$, $y_2 + 1/2\, w_2$, where y_{e1} is equal to the second one of the four values sorted in increasing order and y_{e2} is equal to the third one of the four values sorted in increasing order.

4.2.3 Accessibility between Modules

Since modules are not actual architectural space, they do not have any accessibility to each other, but the corresponding architectural space in the upper part of the module may have. Therefore, accessibility is considered as a relationship between modules, where the real purpose is to study the accessibility of the upper building space. In addition, due to different space unit division schemes for different floors, accessibility is analyzed accordingly.

Regarding 4 types of modules, the accessibility is classified into 4 kinds:

- Accessibility between the tube modules

- Accessibility between the tube module and the corridor module

- Accessibility between the corridor module and the dedicated module, or called, accessibility between the corridor module and the standard space unit.

- Accessibility between the dedicated modules, or called, accessibility within the standard space unit.

For the first kinds of accessibility between two modules, one condition must be meet: the two modules are connected to each other.

For the second and third kinds of accessibility between two modules, one condition must be met: at least one opening on the co-edge of the connected modules. Depending on different relative positions between modules, different types of walls, and the distance of the door to its attaching wall, different conditions to determine the accessibility are illustrated as follows.

- Relative Position: “Down&Up” or “Up&Down”

There are two types of walls: “left drawn walls” and “right drawn walls” on the co-edge.

Firstly, the two types of walls are filtered by the following condition, where the subscript w represents the wall and the subscript $e1$ represents the start point of the co-edge:

$$y_w = y_{e1} \tag{8}$$

Secondly, depending on different wall types, the openings on the co-edge are filtered by different conditions, where the subscript op represents the opening:

“left drawn walls”:

$$x_w - d_{op} \in [x_{e1}, x_{e2}] \tag{9}$$

“right drawn walls”:

$$x_w + d_{op} \in [x_{e1}, x_{e2}] \tag{10}$$

Modules can access each other If at least one opening satisfies the above conditions.

- Relative Position: “Left&Right” or “Right&Left”

There are two types of walls: “down drawn walls” and “up drawn walls” on the co-edge.

Firstly, the two types of walls are further filtered by the following condition:

$$x_w = x_{e1} \tag{11}$$

Secondly, depending on different wall types, the openings on the co-edge are filtered by different conditions:

“down drawn walls”:

$$y_w - d_{op} \in [y_{e1}, y_{e2}] \tag{12}$$

“up drawn walls”:

$$y_w + d_{op} \in [y_{e1}, y_{e2}] \tag{13}$$

Modules can access each other If at least one opening satisfies the above conditions.

For the accessibility within the standard space unit, one of the following conditions must be met: at least one opening is on the co-edge of the connected modules, walls don’t completely cover the co-edge. Depending on different relative positions between modules, different types of walls, and the distance of the door to its attaching wall, different conditions to determine the accessibility are illustrated as follows.

- Relative Position: “Down&Up” or “Up&Down”

There are two types of walls: “left drawn walls” and “right drawn walls” on the co-edge. To fulfil the first condition, equations are the same as the equations (8)-(10). To fulfil the second condition, the steps are as follows:

Firstly, the two types of walls are filtered by the condition same as the equation (10).

Secondly, instead of determine accessibility, inaccessibility is determined by the following different conditions, where the subscript $w1$ represents the left end point of the wall, $w2$ represents the right end point of the wall, $e1$ represents the start point of the co-edge, and $e2$ represents the end point of the co-edge. And notably, w_{min} and w_{max} represents the minimum and maximum wall width, which are user-input data.

If there is only one wall in the filtered results and this wall covers the co-edge between modules:

$$\begin{cases} x_{w1} \leq x_{e1} + \dfrac{w_{min}}{2} \\ x_{w2} \geq x_{e2} - \dfrac{w_{min}}{2} \end{cases} \tag{14}$$

If there is n walls in the filtered results and any one of them covers the co-edge between modules:

$$\begin{cases} x_{w1}^{i} \leq x_{e1} + \dfrac{w_{min}}{2} \\ x_{w2}^{i} \geq x_{e2} - \dfrac{w_{min}}{2} \end{cases} \tag{15}$$

If there is n walls in the filtered results and these walls jointed together to cover the co-edge between modules:

$$\begin{cases} x_{w1}^{1} \leq x_{e1} + \dfrac{w_{min}}{2} \\ x_{w1}^{n} \geq x_{e2} - \dfrac{w_{min}}{2} \\ x_{w1}^{i+1} - x_{w2}^{i} \in (0, w_{max}\ ] \\ x_{w2}^{i} \in \left[x_{e1} + \dfrac{w_{max}}{2}, x_{e2} - \dfrac{w_{max}}{2}\right] \\ x_{w1}^{i+1} \in \left[x_{e1} + \dfrac{w_{max}}{2}, x_{e2} - \dfrac{w_{max}}{2}\right] \end{cases} \tag{16}$$

Thirdly, the two modules can access each other, if none of the cases described in the second step are fulfilled.

- Relative Position: “Left&Right” or “Right&Left”

There are two types of walls: “down drawn walls” and “up drawn walls” on the co-edge. To fulfil the first condition, equations are the same as the equations (11)-(13). To fulfil the second condition, the steps are as follows:

Firstly, the two types of walls are filtered by the condition same as the equation (11).

Secondly, instead of determine accessibility, inaccessibility is determined by the following different conditions:

If there is only one wall in the filtered results and this wall covers the co-edge between modules:

$$\begin{cases} y_{w1}\ \leq y_{e1}\ + \dfrac{w_{min}}{2} \\ y_{w2}\ \geq y_{e2}\ - \dfrac{w_{min}}{2} \end{cases} \tag{17}$$

If there is n walls in the filtered results and any one of them covers the co-edge between modules:

$$\begin{cases} y_{w1}^{i} \leq y_{e1}\ + \dfrac{w_{min}}{2} \\ y_{w2}^{i} \geq y_{e2}\ - \dfrac{w_{min}}{2} \end{cases} \tag{18}$$

If there is n walls in the filtered results and these walls jointed together to cover the co-edge between modules:

$$\begin{cases} y_{w1}^{1} \leq y_{e1} + \dfrac{w_{min}}{2} \\ y_{w1}^{n} \geq y_{e2} - \dfrac{w_{min}}{2} \\ y_{w1}^{i+1} - y_{w2}^{i} \in (0, w_{max}] \\ y_{w2}^{i} \in \left[y_{e1} + \dfrac{w_{max}}{2}, y_{e2} - \dfrac{w_{max}}{2}\right] \\ y_{w1}^{i+1} \in \left[y_{e1} + \dfrac{w_{max}}{2}, y_{e2} - \dfrac{w_{max}}{2}\right] \end{cases} \tag{19}$$

Thirdly, the two modules can access each other, if none of the cases described in the second step are fulfilled.

4.2.4 Building Components Attributed to the Accessibility

Walls on the source floor are subdivided and assigned to the corresponding accessibility between dedicated modules by following three steps. Firstly, the walls are firstly filtered by the source floor elevation. Then, 4 types of walls are filtered out depending on different direction ratios: “left drawn walls”, “right drawn walls”, “up drawn walls”, and “down drawn walls”. After that, different types of walls are further filtered by different conditions under different relative position between the two mutually accessible dedicated modules as follows, where the subscript w represents “wall”, $m1$ represents “module 1”, $m2$ represents “module 2”, $e1$ represents the start point of the co-edge, and $e2$ represents the end point of the co-edge. And notably, the maximum and minimum wall widths w_{w_min} and w_{w_max} are user-input data.

- Relative Position: “Down &Up” or “Up&Down”, Wall Type: “down drawn walls”:

$$\begin{cases} x_w \in (x_{e1} + \dfrac{w_{w\_min}}{2}, x_{e2} - \dfrac{w_{w\_min}}{2}) \\ y_w \in (y_{m1} - \dfrac{w_{m1}}{2} + \dfrac{w_{w\_min}}{2},\ y_{m1} + \dfrac{w_{m1}}{2} + \dfrac{w_{w\_min}}{2}] \\ y_w - l_w \in [y_{m2} - \dfrac{w_{m2}}{2} - \dfrac{w_{w\_min}}{2},\ y_{m2} + \dfrac{w_{m2}}{2} - \dfrac{w_{w\_min}}{2}) \end{cases} \tag{20}$$

or

$$\begin{cases} x_w \in (x_{e1} + \dfrac{w_{w\_min}}{2}, x_{e2} - \dfrac{w_{w\_min}}{2}) \\ y_w \in (y_{m2} - \dfrac{w_{m2}}{2} + \dfrac{w_{w\_min}}{2},\ y_{m2} + \dfrac{w_{m2}}{2} + \dfrac{w_{w\_min}}{2}] \\ y_w - l_w \in [y_{m1} - \dfrac{w_{m1}}{2} - \dfrac{w_{w\_min}}{2},\ y_{m1} + \dfrac{w_{m1}}{2} - \dfrac{w_{w\_min}}{2}) \end{cases} \tag{21}$$

- Relative Position: “Down &Up” or “Up&Down”, Wall Type: “up drawn walls”:

$$\begin{cases} x_w \in (x_{e1} + \frac{w_{w\_min}}{2}, x_{e2} - \frac{w_{w\_min}}{2}) \\ y_w \in [y_{m2} - \frac{w_{m2}}{2} - \frac{w_{w\_min}}{2},\ y_{m2} + \frac{w_{m2}}{2} - \frac{w_{w\_min}}{2}) \\ y_w + l_w \in (y_{m1} - \frac{w_{m1}}{2} + \frac{w_{w\_min}}{2},\ y_{m1} + \frac{w_{m1}}{2} + \frac{w_{w\_min}}{2}] \end{cases} \quad (22)$$

or

$$\begin{cases} x_w \in (x_{e1} + \frac{w_{w\_min}}{2}, x_{e2} - \frac{w_{w\_min}}{2}) \\ y_w \in [y_{m1} - \frac{w_{m1}}{2} - \frac{w_{w\_min}}{2},\ y_{m1} + \frac{w_{m1}}{2} - \frac{w_{w\_min}}{2}) \\ y_w + l_w \in (y_{m2} - \frac{w_{m2}}{2} + \frac{w_{w\_min}}{2},\ y_{m2} + \frac{w_{m2}}{2} + \frac{w_{w\_min}}{2}] \end{cases} \quad (23)$$

- Relative Position: “Down &Up” or “Up&Down”, Wall Type: “left drawn walls”:

$$\begin{cases} x_w \in (x_{e1} + \frac{w_{w\_min}}{2}, x_{e2} + \frac{w_{w\_min}}{2}] \\ y_w = y_{e1}\ \ or\ \ y_w = y_{e2} \end{cases} \quad (24)$$

or

$$\begin{cases} x_w - l_w \in [x_{e1} - \frac{w_{w\_min}}{2}, x_{e2} - \frac{w_{w\_min}}{2}) \\ y_w = y_{e1}\ \ or\ \ y_w = y_{e2} \end{cases} \quad (25)$$

- Relative Position: “Down &Up” or “Up&Down”, Wall Type: “right drawn walls”:

$$\begin{cases} x_w \in [x_{e1} - \frac{w_{w\_min}}{2}, x_{e2} - \frac{w_{w\_min}}{2}) \\ y_w = y_{e1}\ \ or\ \ y_w = y_{e2} \end{cases} \quad (26)$$

or

$$\begin{cases} x_w + l_w \in (x_{e1} + \frac{w_{w\_min}}{2}, x_{e2} + \frac{w_{w\_min}}{2}] \\ y_w = y_{e1}\ \ or\ \ y_w = y_{e2} \end{cases} \quad (27)$$

- Relative Position: “Left &Right” or “Right&Left”, Wall Type: “down drawn walls”:

$$\begin{cases} x_w = x_{e1}\ \ or\ \ x_w = x_{e2} \\ y_w \in (y_{e1} + \frac{w_{w\_min}}{2}, y_{e2} + \frac{w_{w\_min}}{2}] \end{cases} \quad (28)$$

or

$$\begin{cases} x_w = x_{e1}\ \ or\ \ x_w = x_{e2} \\ y_w - l_w \in [y_{e1} - \frac{w_{w\_min}}{2}, y_{e2} - \frac{w_{w\_min}}{2}) \end{cases} \quad (29)$$

- Relative Position: “Left &Right” or “Right&Left”, Wall Type: “up drawn walls”:

$$\begin{cases} x_w = x_{e1} \ \ or \ \ x_w = x_{e2} \\ y_w \in [y_{e1} - \frac{w_{w\_min}}{2}, y_{e2} - \frac{w_{w\_min}}{2}) \end{cases} \tag{30}$$

or

$$\begin{cases} x_w = x_{e1} \ \ or \ \ x_w = x_{e2} \\ y_w + \ l_w \in (y_{e1} + \frac{w_{w\_min}}{2}, y_{e2} + \frac{w_{w\_min}}{2}] \end{cases} \tag{31}$$

- Relative Position: “Left &Right” or “Right&Left”, Wall Type: “left drawn walls”:

$$\begin{cases} x_w \in (x_{m1} - \frac{l_{m1}}{2} + \frac{w_{w\_min}}{2}, x_{m1} + \frac{l_{m1}}{2} + \frac{w_{w\_min}}{2}] \\ x_w - \ l_w \in [x_{m2} - \frac{l_{m2}}{2} - \frac{w_{w\_min}}{2}, x_{m2} + \frac{l_{m2}}{2} - \frac{w_{w\_min}}{2}) \\ y_w \in \ (y_{e1} + \frac{w_{w\_min}}{2}, y_{e2} - \frac{w_{w\_min}}{2}) \end{cases} \tag{32}$$

or

$$\begin{cases} x_w \in (x_{m2} - \frac{l_{m2}}{2} + \frac{w_{w\_min}}{2}, x_{m2} + \frac{l_{m2}}{2} + \frac{w_{w\_min}}{2}] \\ x_w - \ l_w \in [x_{m1} - \frac{l_{m1}}{2} - \frac{w_{w\_min}}{2}, x_{m1} + \frac{l_{m1}}{2} - \frac{w_{w\_min}}{2}) \\ y_w \in \ (y_{e1} + \frac{w_{w\_min}}{2}, y_{e2} - \frac{w_{w\_min}}{2}) \end{cases} \tag{33}$$

- Relative Position: “Left &Right” or “Right&Left”, Wall Type: “right drawn walls”:

$$\begin{cases} x_w \in [x_{m1} - \frac{l_{m1}}{2} - \frac{w_{w\_min}}{2}, x_{m1} + \frac{l_{m1}}{2} - \frac{w_{w\_min}}{2}) \\ x_w + \ l_w \in (x_{m2} - \frac{l_{m2}}{2} + \frac{w_{w\_min}}{2}, x_{m2} + \frac{l_{m2}}{2} + \frac{w_{w\_min}}{2}] \\ y_w \in \ (y_{e1} + \frac{w_{w\_min}}{2}, y_{e2} - \frac{w_{w\_min}}{2}) \end{cases} \tag{34}$$

or

$$\begin{cases} x_w \in [x_{m2} - \frac{l_{m2}}{2} - \frac{w_{w\_min}}{2}, x_{m2} + \frac{l_{m2}}{2} - \frac{w_{w\_min}}{2}) \\ x_w + \ l_w \in (x_{m1} - \frac{l_{m1}}{2} + \frac{w_{w\_min}}{2}, x_{m1} + \frac{l_{m1}}{2} + \frac{w_{w\_min}}{2}] \\ y_w \in \ (y_{e1} + \frac{w_{w\_min}}{2}, y_{e2} - \frac{w_{w\_min}}{2}) \end{cases} \tag{35}$$

4.3 Graph Representation of Building design

After data extraction and processing, a property graph can be modelled. A property graph includes three main elements: nodes, relationships, and their properties. The correspondence between the building-related data and the graph elements in this graph data model are as follows:

- Nodes: Modules

- Node Properties: Module attributes
- Relationships: Connectivity and accessibility between modules
- Relationship Properties: Connectivity attributes and accessibility attributes

The configurations for each type of graph data are described in the following subsections.

4.3.1 Graph Node and Node Properties

4 types of modules correspond to 4 types of graph nodes. The configuration of all graph nodes is illustrated in Table 4-1.

Table 4-1 Graph Node Configuration

Node Label	Node Property
Tube_Module	global_id, type_name = “tube_module”
Corridor_Module	global_id, type_name= “corridor_module”
Basic_ Module	global_id, type_name = “basic_module”, components
Conjunctive_Module	global_id, type_name = “conjunctive_module”, components

As illustrated in Table 4-1, tube modules are labelled as “Tube_Module”, corridor modules are labelled as “Corridor_Module”, basic modules are labelled as “Basic Module”, and conjunctive modules are labelled as “Conjunctive_Module”. All nodes have two properties: “global_id” and “type_name”, and nodes with the label “Basic_Module” and “Conjunctive_Module” have one more property: “components”.

Three types of node properties are defined as follows:

- global_id: IFC_GUID of the module. It serves as the primary key of the node.

- type_name: the type of the module. It is used as the node caption to enhance legibility in the graph database console.

- components: the IFC_GUIDS list of building components under the dedicated module, including slabs, furnishing elements, assembled rooms, and walls with attached doors.

The type and example value of the node properties are illustrated in Table 4-2.

Table 4-2 Graph Node Properties

Node Property	Type	Value
global_id	string	e.g., 0Z3tKe7qv7jOUiPp0aCHt6
type_name	string	tube_module/corridor_module/ basic_module/conjunctive_module
components	string []	e.g., [250l3GdfrCxOtkrZjNE2EX, 2t746mC715Hu6lDCInoumy, …]

4.3.2 Graph Relationship and Relationship Properties

There are three relationship types in this graph data model:

- the connectivity between corresponds to a graph relationship type.
- the accessibility within the source floor corresponds to a graph relationship type.
- the accessibility within the objective floor corresponds to a graph relationship type.

Regarding different relationship types and different attached nodes, the graph relationship configuration is illustrated in Table 4-3.

Three types of relationship properties are defined as follows:

- angle: A directed angle is formed by counterclockwise rotation from the positive direction of the x-axis to the directed line segment, starting at the centre point of the basic module and ending at the centre point of the conjunctive module. It is used to determine the transformation type from one source standard functional unit to one corresponding objective standard functional unit.

- door_family_type: It is the family type of door that can access the corridor from the standard function unit or the standard function unit from the corridor. It is used to match the source standard functional unit and the objective standard functional unit.

- components: This is the IFC_GUID list of walls with attached doors under the accessibility between dedicated modules within the source floor.

Table 4-3 Graph Relationship Configuration

Relationship: CONNECTS		
Start Node	**End Node**	**Relationship Property**
Tube_Module	Tube_Module	
Tube_Module	Corridor_Module	
Tube_Module	Basic_ Module	
Tube_Module	Conjunctive_Module	
Corridor_Module	Corridor_Module	
Corridor_Module	Basic_ Module	
Corridor_Module	Conjunctive_Module	
Basic_ Module	Basic_ Module	
Basic_ Module	Conjunctive_Module	angle
Conjunctive_Module	Conjunctive_Module	
Relationship: ACCESSES_1		
Start Node	**End Node**	**Relationship Property**
Tube_Module	Tube_Module	
Tube_Module	Corridor_Module	
Corridor_Module	Corridor_Module	
Corridor_Module	Basic_Module	door_family_type
Corridor_Module	Conjunctive_Module	door_family_type
Basic_Module	Conjunctive_Module	components
Conjunctive_Module	Conjunctive_Module	components
Relationship: ACCESSES_2		
Start Node	**End Node**	**Relationship Property**
Tube_Module	Tube_Module	
Tube_Module	Corridor_Module	
Corridor_Module	Corridor_Module	
Corridor_Module	Basic_Module	door_family_type
Corridor_Module	Conjunctive_Module	door_family_type
Basic_Module	Conjunctive_Module	
Conjunctive_Module	Conjunctive_Module	

In Table 4-3, the relationship “CONNETS” between the node “Basic_Module” and the node “Conjunctive_Module” has one property: “angle”. 2 relationships have the property “components”, which are the relationship “ACCESSES_1” between the node “Basic_Module” and the node “Conjunctive_Module” and the relationship “ACCESSES_1” between two “Conjunctive_Module” nodes. 4 relationships have the property “door_family_type”, which are the relationship “ACCESSES_1” between the node “Corridor_Module” and the node “Basic_Module”, the relationship “ACCESSES_1” between the node “Corridor_Module” and the node “Conjunctive_Module”, the relationship “ACCESSES_2” between the node “Corridor_Module” and the node “Basic_Module”, and the relationship “ACCESSES_2” between the node “Corridor_Module” and the node “Conjunctive_Module”

The type and example value of the relationship properties are illustrated in Table 4-4.

Table 4-4 Graph Relationship Property

Node Property	Type	Value
angle	float	angle ∈ $(-180°, 180°]$ e.g., - 120.07°
door_family_type	string	e.g., ZM1121_1
components	string []	e.g., [250l3GdfrCxOtkrZjNE2EX, 2t746mC715Hu6lDCInou, …]

4.4 Data Retrieval with Graph Database

Data retrieval from the database is a pivotal step in realizing design automation. For the proposed modularized design process for prefabricated building floorplans featuring frame-tube structures, the greater the number of modules, the more intricate the relationships between them become. This complexity results in cumbersome tasks related to data querying. The graph database excels particularly in handling intricate relationships among entities. Consequently, with the graph database, a thorough traversal of the relationships between modules can be provided to retrieve the needed datasets precisely and efficiently.

4.4.1 Data Queries for the Source Floor

According to the space unit division schema of the source floor, 2 queries (Query 1 and Query 2) are performed to obtain the following 3 types of data. The process is illustrated as Figure 4.8.

- All modules of the standard space unit for each source standard functional unit.
- All building components of the internal layout for each source standard functional unit.
- All modules of the standard space unit for each objective standard functional unit on the source floor.

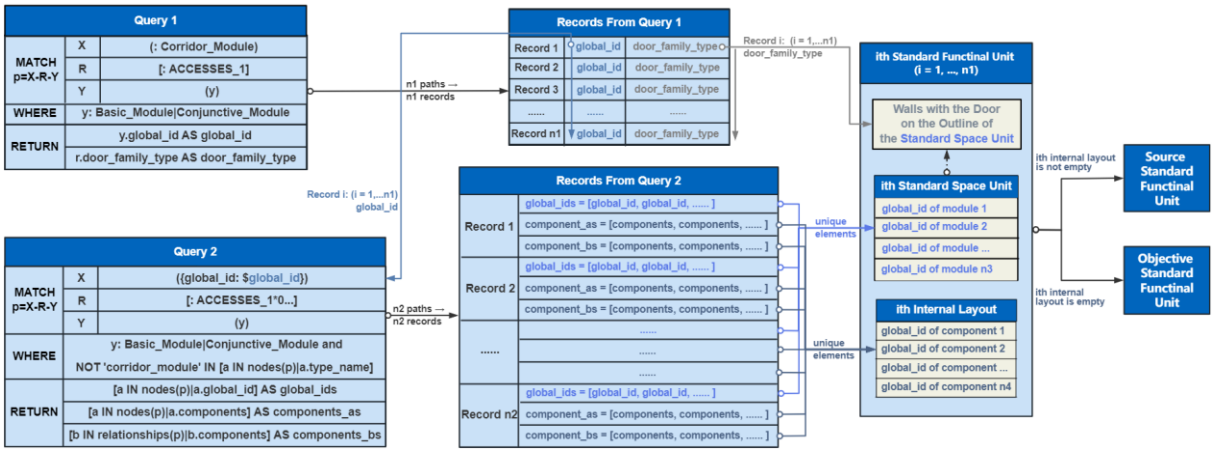

Figure 4.8. Data Retrieval based on Query 1 and Query 2

Query 1: Query all the paths with the relationship "ACCESSES_1" between a “Corridor_Module” node and a node, where the node is either “Basic_Module” or “Conjunctive_Module”. The query returns one record for each path that includes the property "global_id" of the node and the property "door_family_type" of the relationship.

In Query 1, the returned node is one of the modules that makes up one standard space unit, while the returned property “door_family_type” is the family type of the door, which is attached to the wall on the outline of one corresponding standard functional unit. Generally, if the number of an object is defined by the symbol N_{object}, the quantitative relationship between the above related objects is illustrated below:

$$N_{record} = N_{returned\_node} = N_{door\_family\_type} = N_{standard\_space\_unit} = N_{standard\_functional\_unit}$$

Query 2: Query all the paths which contain the node with the property “global_id” obtained from Query 1. On each path, all relationships are "ACCESSES_1" and each node is either “Basic_Module” or “Conjunctive_Module”. The query returns one record

for each path that includes the property "global_id" for all nodes, the property "component" for all nodes, and the property "component" for all relationships.

Query 2 is looped until all nodes in Query 1 are utilized. Aggregating all paths of one query, all unique nodes form one entire standard space unit, while all unique building components make up the internal layout of one source standard functional unit after data processing of the non-empty component sets. Notably, if all component sets are empty, the standard functional unit is an objective unit. If not, the standard functional unit is a source unit.

From all modules of the standard space unit for each source standard functional unit, the basic module can be filtered out. Then 2 queries (Query 3 and Query 4) are performed to obtain the following type of data. The process is illustrated as Figure 4.9.

- For the standard space unit of each source standard functional unit on the source floor, the property "angle" of the connectivity relationship between the basic module and its accessible conjunctive modules.

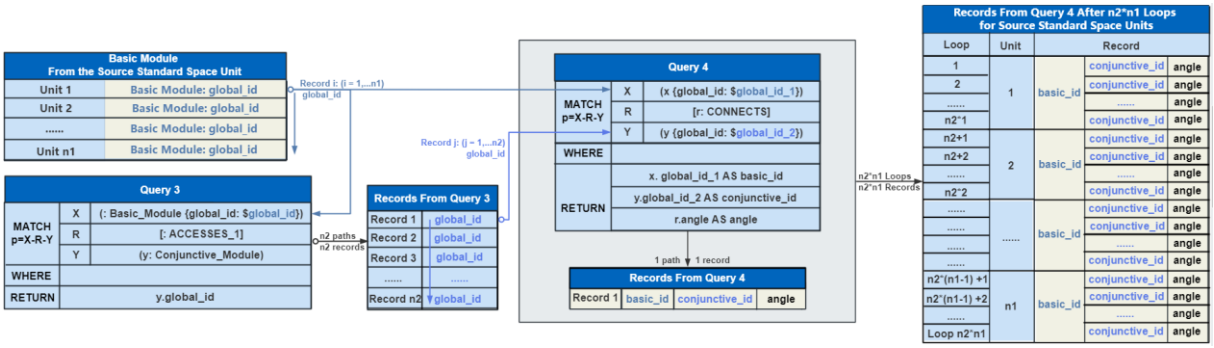

Figure 4.9. Data Retrieval based on Query 3 and Query 4

Query 3: Query all the paths with the relationship "ACCESSES_1" between a “Basic_Module” node and a “Conjunctive_Module” node, where the property “global_id” of the “Basic_Module” node is specified from a standard space unit of the source standard functional unit (called the source standard space unit in Figure 4.9.). The query returns one record for each relationship that includes the property “global_id” of the “Conjunctive_Module” node.

Query 3 is looped until all “Basic_Module” nodes of source standard space units are utilized. By performing the query once, all “Conjunctive_Module” nodes accessed to the basic module within one standard space unit can be retrieved.

Query 4: Query one path with the relationship “CONNECTS” between a “Basic_Module” node and a “Conjunctive_Module” node, where the “Basic_Module”

node is same as the one in Query 3, and the “Conjunctive_Module” node is same as the one of each returned record after Query 3. The query returns one record for the path that includes the property “global_id” of the “Basic_Module” node, the property “global_id” of the “Conjunctive_Module” node, and the property “angle” of the relationship.

Query 4 is embedded within a dual-layered loop structure. The number of iterations in the outer loop is determined by the number of basic modules, while the number of iterations in the inner loop is determined by the number of returned records from Query 3. After all loops, the angle between the basic module and any conjunctive module connected to it within each source standard space unit are obtained.

From all modules of the standard space unit for each objective standard functional unit of the source floor, the basic module can be filtered. Then 2 queries (Query 5 and Query 6) are performed to obtain the following type of data. The process is illustrated as Figure 4.10.

- For the standard space unit of each objective standard functional unit on the source floor, the property **"**angle**"** of the connectivity relationship between the basic module and its accessible conjunctive modules.

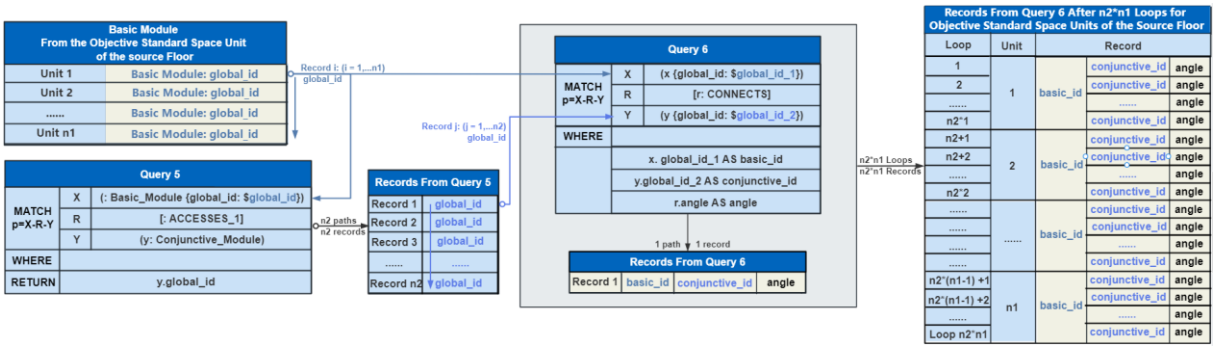

Figure 4.10. Data Retrieval Based on Query 5 and Query 6

Query 5: Query all the paths with the relationship "ACCESSES_1" between a “Basic_Module” node and a “Conjunctive_Module” node, where the property “global_id” of the “Basic_Module” node is specified from a standard space unit of the objective standard functional unit (called the objective standard space unit in Figure 4.10) on the source floor. The query returns one record for each relationship that includes the property “global_id” of the “Conjunctive_Module” node.

Query 5 is looped until all “Basic_Module” nodes of the objective standard space units are utilized. By performing the query once, all “Conjunctive_Module” nodes accessed

to the basic module within one standard space unit can be retrieved.

Query 6: Query one path with the relationship “CONNECTS” between a “Basic_Module” node and a “Conjunctive_Module” node, where the “Basic_Module” node is same as the one in Query 5, and the “Conjunctive_Module” node is same as the one of each returned record after Query 5. The query returns one record for the path that includes the property “global_id” of the “Basic_Module” node, the property “global_id” of the “Conjunctive_Module” node, and the property “angle” of the relationship.

Query 6 is embedded within a dual-layered loop structure. The number of iterations in the outer loop is determined by the number of basic modules, while the number of iterations in the inner loop is determined by the number of returned records from Query 5. After all loops, the angle between the basic module and any conjunctive module connected to it within each objective standard space unit of the source floor are obtained.

4.4.2 Data Queries for the Objective Floor

According to the space unit division schema of the objective floor, 2 queries (Query 7 and Query 8) are performed to obtain the following type of data. The process is illustrated as Figure 4.11.

- All modules of the standard space unit for each objective standard functional unit on the objective floor.

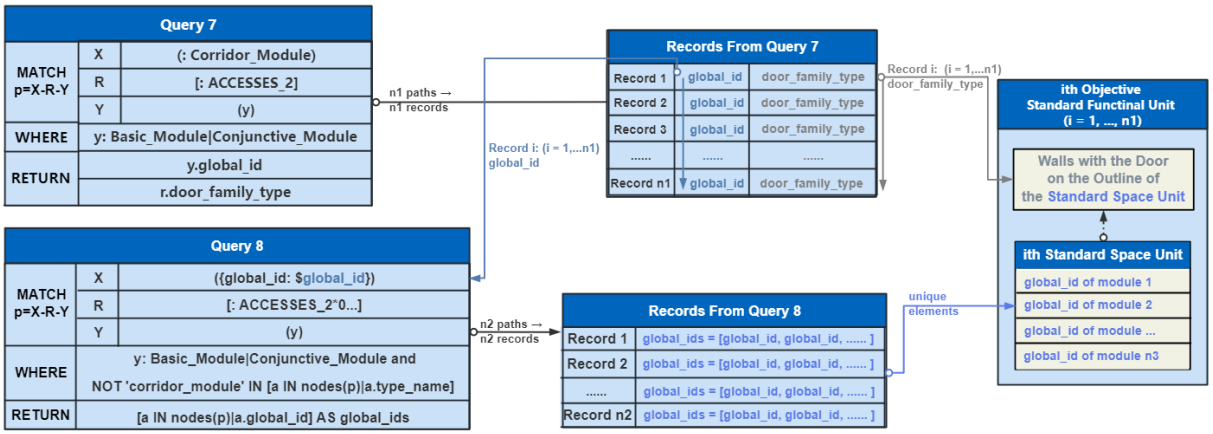

Figure 4.11. Data Retrieval based on Query 7 and Query 8

Query 7: Query all the paths with the relationship "ACCESSES_2" between a “Corridor_Module” node and a node, where the node is either “Basic_Module” or “Conjunctive_Module”. The query returns one record for each path that includes the

property "global_id" of the node and the property "door_family_type" of the relationship.

In Query 7, the returned node is one of the modules that makes up one standard space unit, while the returned property “door_family_type” is the family type of the door, which is attached to the wall on the outline of one corresponding standard functional unit. Generally, if the number of an object is defined by the symbol N_{object}, the quantitative relationship between the above related objects is illustrated below:

$$N_{record} = N_{returned\_node} = N_{door\_family\_type} = N_{standard\_space\_unit} = N_{standard\_functional\_unit}$$

Query 8: Query all the paths which contain the node with the property “global_id” obtained from Query 7. On each path, all relationships are "ACCESSES_2" and each node is either “Basic_Module” or “Conjunctive_Module”. The query returns one record for each path that includes the property "global_id" for all nodes.

Query 8 is looped until all nodes in Query 7 are utilized. Aggregating all paths of one query, all unique nodes together form one entire standard space unit. Since all standard function units on the objective floor lack their internal layout, the units are objective units.

From all modules of the standard space unit for each objective standard functional unit of the objective floor, the basic module can be filtered. Then 2 queries (Query 9 and Query 10) are performed to obtain the following type of data. The process is illustrated as Figure 4.12.

- For the standard space unit of each objective standard functional unit on the objective floor, the property "angle" of the connectivity relationship between the basic module and its accessible conjunctive modules.

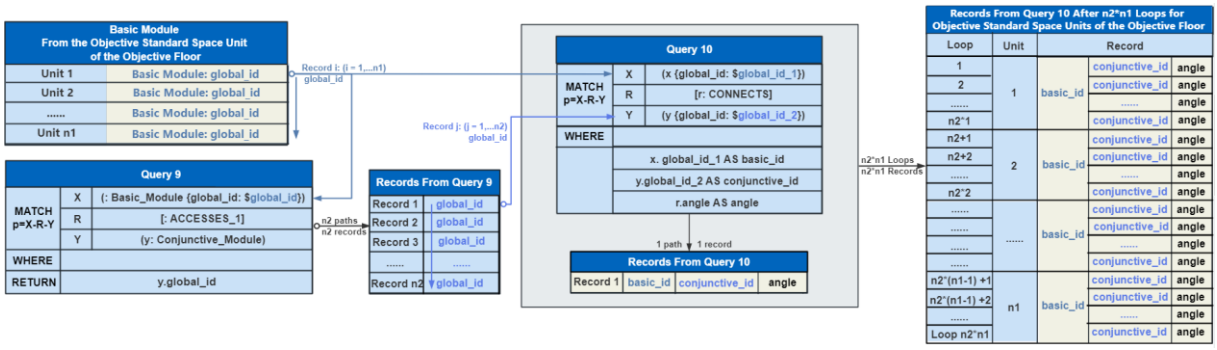

Figure 4.12. Data Retrieval Based on Query 9 and Query 10

Query 9: Query all the paths with the relationship "ACCESSES_2" between a “Basic_Module” node and a “Conjunctive_Module” node, where the property “global_id”

of the “Basic_Module” node is specified from a standard space unit of the objective standard functional unit (called the objective standard space unit in Figure 4.12.) on the objective floor. The query returns one record for each relationship that includes the property “global_id” of the “Conjunctive_Module” node.

Query 9 is looped until all “Basic_Module” nodes of the objective standard space units are utilized. By performing the query once, all “Conjunctive_Module” nodes accessed to the basic module within one standard space unit can be retrieved.

Query 10: Query one path with the relationship “CONNECTS” between a “Basic_Module” node and a “Conjunctive_Module” node, where the “Basic_Module” node is same as the one in Query 9, and the “Conjunctive_Module” node is same as the one of each returned record after Query 9. The query returns one record for the path that includes the property “global_id” of the “Basic_Module” node, the property “global_id” of the “Conjunctive_Module” node, and the property “angle” of the relationship.

Query 10 is embedded within a dual-layered loop structure. The number of iterations in the outer loop is determined by the number of basic modules, while the number of iterations in the inner loop is determined by the number of returned records from Query 9. After all loops, the angle between the basic module and any conjunctive module connected to it within each objective standard space unit of the objective floor are obtained.

4.5 Data Transformation

4.5.1 Transformation Object

The objects of transformation are internal layouts of all source standard functional units. Each Internal layout consists of slabs, furnishing elements, assembled rooms, and walls with attached doors. The objects can be automatically obtained by following the steps in the previous sections.

4.5.2 Transformation Type

The internal layout of a source standard functional unit can be transformed into an objective standard functional unit if the doors attached to the wall on the outline of the source standard functional unit and the objective standard functional unit belong to the same door family type.

The automated design process in this study considers the following basic transformations:

- Translation
- Vertical Mirror
- Horizontal Mirror
- 90-degree Counterclockwise Rotation
- 180-degree Counterclockwise Rotation
- 270-degree Counterclockwise Rotation

From a source unit to an objective unit, the actual transformation may also be a combination of the basic transformations listed above. Since some different combinations lead to the same result, e.g., the combination “a vertical mirror after a 270-degree counterclockwise rotation” is equivalent to the combination “a horizontal mirror after a 90-degree counterclockwise rotation”, only one of them is selected. Eventually, all possible transformations considered in this study are as follows:

- Translation
- Translation + Vertical Mirror
- Translation + Horizontal Mirror
- Translation + 90-degree Counterclockwise Rotation
- Translation + 180-degree Counterclockwise Rotation
- Translation + 270-degree Counterclockwise Rotation
- Translation + 90-degree Counterclockwise Rotation + Vertical Mirror
- Translation + 90-degree Counterclockwise Rotation + Horizontal Mirror

Notably, “no translation” is regarded as a translation where the translation vector has no length.

4.5.3 Transformation Rule

The internal layout of a source standard functional unit can be transformed into an objective standard functional unit if the doors attached to the wall on the outline of the source standard functional unit and the objective standard functional unit belong to the same door family type.

The transformation rules are established to determine the transformation type from the source standard space unit to the objective standard space unit. The property “angle”

of the connectivity between the basic module and any accessible conjunctive module within the unit is the key constraint variable for the transformation rules.

Within a unit, a basic module may access one or more conjunctive modules, which also implies the existence of one or more related connectivity. On the premise that the internal layout of a known source unit is able to be transformed into a known objective unit if the properties “angle” under the related connectivity in the source unit are included in set X and the properties “angle” under the related connectivity in the objective unit are included in set Y, then there must be a bijective function between set X and set Y such that each element of either set is paired with exactly one element of the other set. The mathematical expression is as follows:

$$\forall\, x \in A, \exists!\, y \in B \rightarrow\ f(x) = y \tag{36}$$

$$\forall\, y \in B, \exists!\, x \in A \rightarrow\ f^{-1}(y) = x \tag{37}$$

Specifically, different transformations lead to different bijective functions.

If the transformation from a known source unit to a known objective unit is a “Translation”, the bijection function is as follows:

$$f(x) = x, \qquad x \in (-180, 180] \tag{38}$$

If the transformation from a known source unit to a known objective unit is “Translation + Horizontal Mirror”, the bijection function is as follows:

$$f(x) = \begin{cases} -x - 180, x \in (-180, 0) \\ -x + 180,\ \ x \in [0, 180] \end{cases} \tag{39}$$

If the transformation from a known source unit to a known objective unit is a “Translation + Vertical Mirror”, the bijection function is as follows:

$$f(x) = \begin{cases} -x, x \in (-180, 180) \\ 180,\ \ x = 180 \end{cases} \tag{40}$$

If the transformation from a known source unit to a known objective unit is a “Translation + 90-degree Counterclockwise Rotation”, the bijection function is as follows:

$$f(x) = \begin{cases} x + 90, x \in (-180, 90] \\ x - 270,\ \ x \in (90, 180) \end{cases} \tag{41}$$

If the transformation from a known source unit to a known objective unit is a “Translation + 180-degree Counterclockwise Rotation”, the bijection function is as follows:

$$f(x) = \begin{cases} x + 180, x \in (-180, 0] \\ x - 180, \ \ x \in (0, 180] \end{cases} \quad (42)$$

If the transformation from a known source unit to a known objective unit is a "Translation + 270-degree Counterclockwise Rotation", the bijection function is as follows:

$$f(x) = \begin{cases} x + 270, \quad\ \ x \in (-180, -90] \\ x - 90, \ \ x \in (-90, 180] \end{cases} \quad (43)$$

If the transformation from a known source unit to a known objective unit is a "Translation + 90-degree Counterclockwise Rotation + Horizontal Mirror", the bijection function is as follows:

$$f(x) = \begin{cases} -x - 270, x \in (-180, -90) \\ -x + 90, \ \ x \in [-90, 180] \end{cases} \quad (44)$$

If the transformation from a known source unit to a known objective unit is a "Translation + 90-degree Counterclockwise Rotation + Vertical Mirror", the bijection function is as follows:

$$f(x) = \begin{cases} -x - 90, x \in (-180, 90) \\ -x + 270, \ \ x \in [90, 180] \end{cases} \quad (45)$$

According to the previous definition of angle, the interval of the independent variable x is always between (-180, 180] in the above bijective functions. Furthermore, in bijective functions, the function in each interval is regarded as a transformation rule, expressed uniformly as $f(x) = kx + b$, where k is the slope, and b is the y-intercept. By using this uniform expression, all the transformation rules are illustrated in Table 4-5.

Notably, if there is only one basic module without any conjunctive module within the transformed unit, the transform type can be any type listed above.

4.5.4 Transformation Reference Points

To realize the transformation of an internal layout of the source unit, besides determining the type of transformation from the source unit to the matching objective unit according to the transformation rules, the reference points for the transformation need to be determined. Reference points are a pair of two points that provide a reference for the transformation.

In this thesis, the centre points of basic modules are used as reference units. Since the basic module of each unit is unique, the reference points can be identified after

matching the source unit and objective unit.

Transformation operations, including rotation, translation, and mirroring, can be performed automatically from the source unit to the objective unit according to the positional coordinates of the reference.

Table 4-5. Transformation Rules

Angle(x) From Source Unit	Angle(y) From Objective Unit y = kx+b		Transformation Type
	k	b	
$x \in (-180, 180]$	1	0	Translation
$x \in (-180, 0)$	-1	-180	Translation + Horizontal Mirror
$x \in [0, 180]$	-1	180	
$x \in (-180, 180)$	-1	0	Translation + Vertical Mirror
$x = 180$	0	180	
$x \in (-180, 90]$	1	90	Translation + 90-degree Counterclockwise Rotation
$x \in (90, 180]$	1	-270	
$x \in (-180, 0]$	1	180	Translation + 180-degree Counterclockwise Rotation
$x \in (0, 180]$	1	-180	
$x \in (-180, -90]$	1	270	Translation + 270-degree Counterclockwise Rotation
$x \in (-90, 180]$	1	-90	
$x \in (-180, -90)$	-1	-270	Translation + 90-degree Counterclockwise Rotation + Horizontal Mirror
$x \in [-90, 180]$	-1	90	
$x \in (-180, 90)$	-1	-90	Translation + 90-degree Counterclockwise Rotation + Vertical Mirror
$x \in [90, 180]$	-1	270	

5 Case Study and Result

The case study validates the proposed modularized design and the design automation based on the automated design workflow. The case is a high-rise prefabricated residential building project with a frame-tube structure in the design phase. The main steps of the case study are illustrated below:

- Design the preliminary floorplans for this case based on the modularized design presented in Chapter 3.

- Create graph data models by transiting building information into graph information regarding Chapter 4.1, Chapter 4.2, and Chapter 4.3.

- Generate a complete floor plan generation scheme as a file in .csv format regarding Chapter 4.4 and Chapter 4.5.

- Generate a Revit Addin using Revit API under C# Project and implement the automated transformation within Revit Model regarding Chapter 4.5.

5.1 Preliminary Floorplan Design

The BIM authoring tool Autodesk Revit 2023 is used to configure floor plans for the project. Based on the modularization proposed in the previous chapter, the initial design task is to configure modules for the project. As illustrated in Table 5-1, modules are configured by different types, colours, dimensions, and quantities.

Table 5-1. Module Configurations

Module Type		Module Color	Module Dimensions (mm)			Quantity
			Length	Width	Height	
Tube Module		Grey	7500	6900	3000	1
Corridor Module		Light Beige	11400	2400	3000	1
			4500	4500	3000	1
Dedicated Modules	Basic Module	Dark Blue	6900	6900	3000	6
	Conjunctive Module	Light Blue	4500	4500	3000	3

After the module configuration, space units are designed by one module or combining modules of the same type. In this case, the tube space unit is formed by one tube module, while two corridor space units are formed by two corridor modules of different sizes each. The key point of the space unit configuration is multiple standard space units that consist of dedicated modules. The library of standard space units and their variants is illustrated in Figure 5.1.

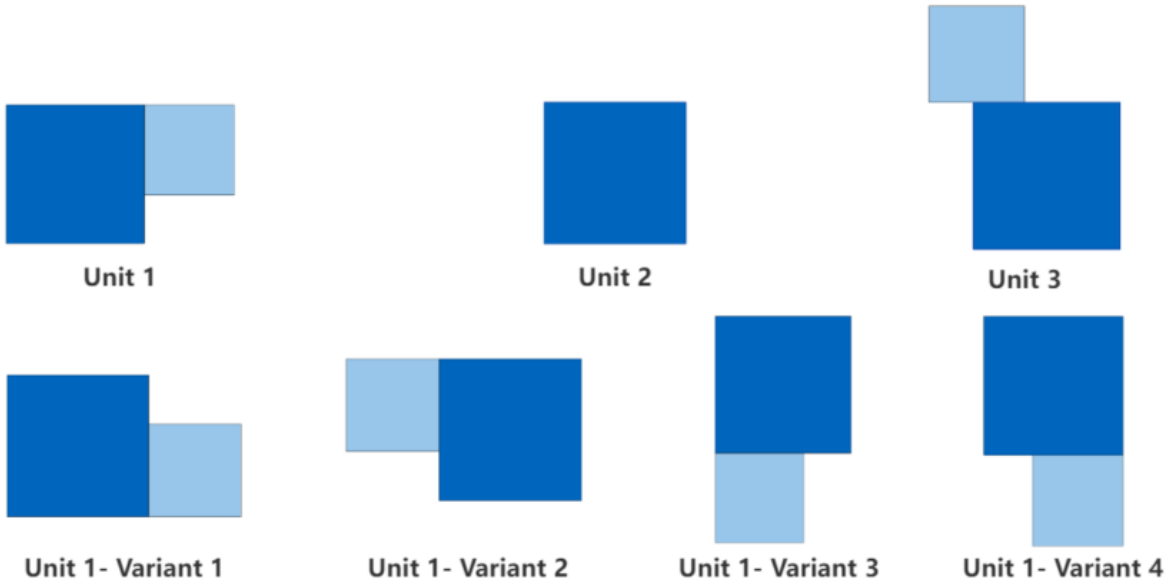

Figure 5.1. Standard Space Unit Library

After the standard space unit configuration, the modularized floor layout is illustrated in Figure 5.2.

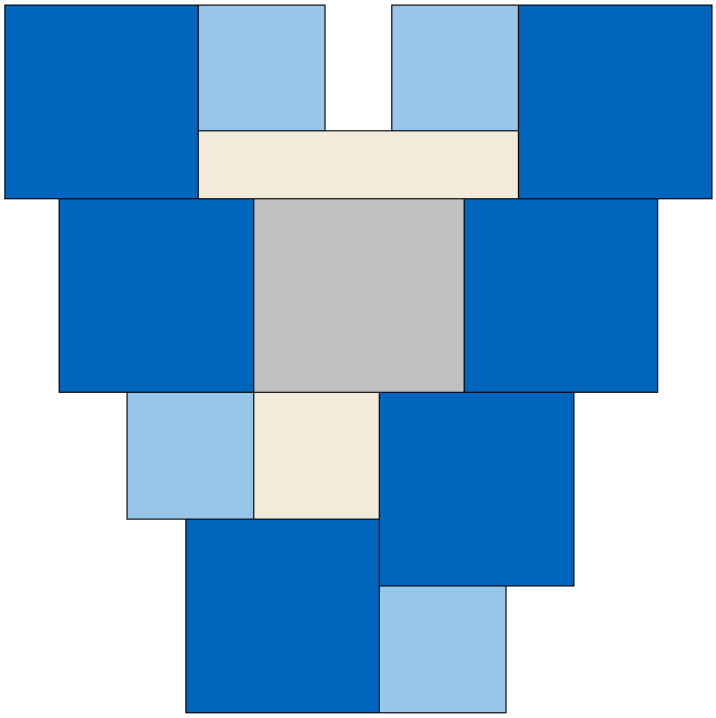

Figure 5.2 Modularized Floor Plan

Notably, the modularized floor plan is designed on a separate storey, which is lower than all the house plans. The tube module is placed at or near the centre within the floor plan, and the two corridor modules are sandwiched between the dedicated modules and the tube module. Further, the modularized layout is refined by combinations of different standard space units selected from the standard space unit library. Therefore, based on the modularized layout, the space unit division schemes both for the source floor and for the objective floor are illustrated in Figure 5.3.

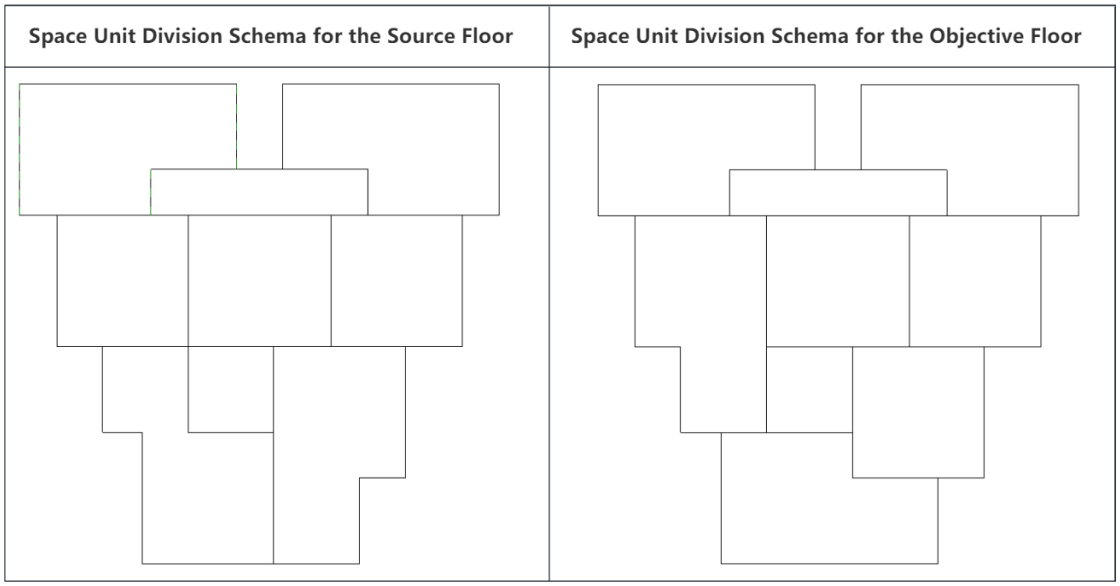

Figure 5.3. Space Unit Division Schemes

For the two space unit division schemes, the occurrences of different standard space units selected from the library are illustrated in Table 5-2.

Table 5-2. Occurrences of Different Standard Space Units per Schema

Standard Space Unit	Occurrences of Different Standard Space Units per Schema	
Type	For the Source Floor	For the Objective Floor
Unit 1	1	1
Unit 1 - Variant 1	0	1
Unit 1 - Variant 2	1	1
Unit 1 - Variant 3	1	0
Unit 1 - Variant 4	0	1
Unit 2	2	2
Unit 3	1	0

Table 5-2 shows that at least one unit is selected as a source standard space unit from the Unit 1 type and its variants under the unit space division scheme for the source floor, while at least one unit is selected as a source standard space unit from the Unit 2 type. Since the unit under the Unit 3 type is unique and there is no variant of this type, it is a source standard space unit that does not involve automated transformations. After all source units are identified, the remaining standard space units on the source floor and all standard space units on the objective floor are all objective units.

With all kinds of space units, the corresponding component-level units can be generated. The configuration of the tube unit and corridor unit on the source and objective floors is illustrated in Figure 5.4.

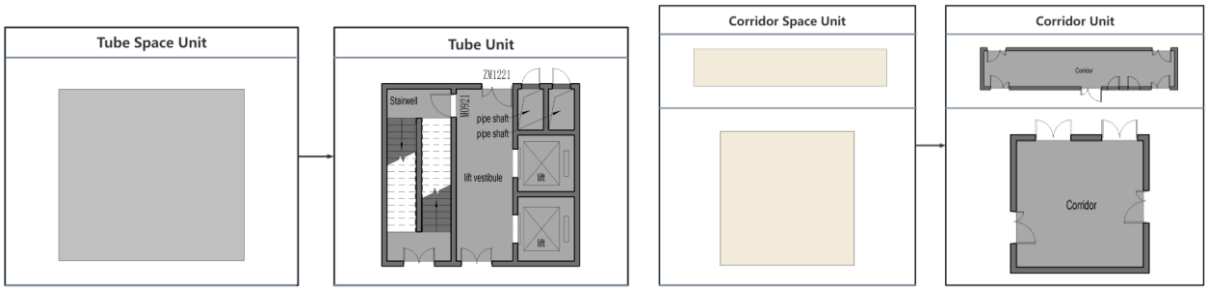

Figure 5.4. Configuration of the tube unit and corridor

As shown in Figure 5.4, the tube space unit corresponds to the tube unit, while 2 corridor space units correspond to 2 corridor units. The configuration of all source standard functional units on the source floor is illustrated in Figure 5.5.

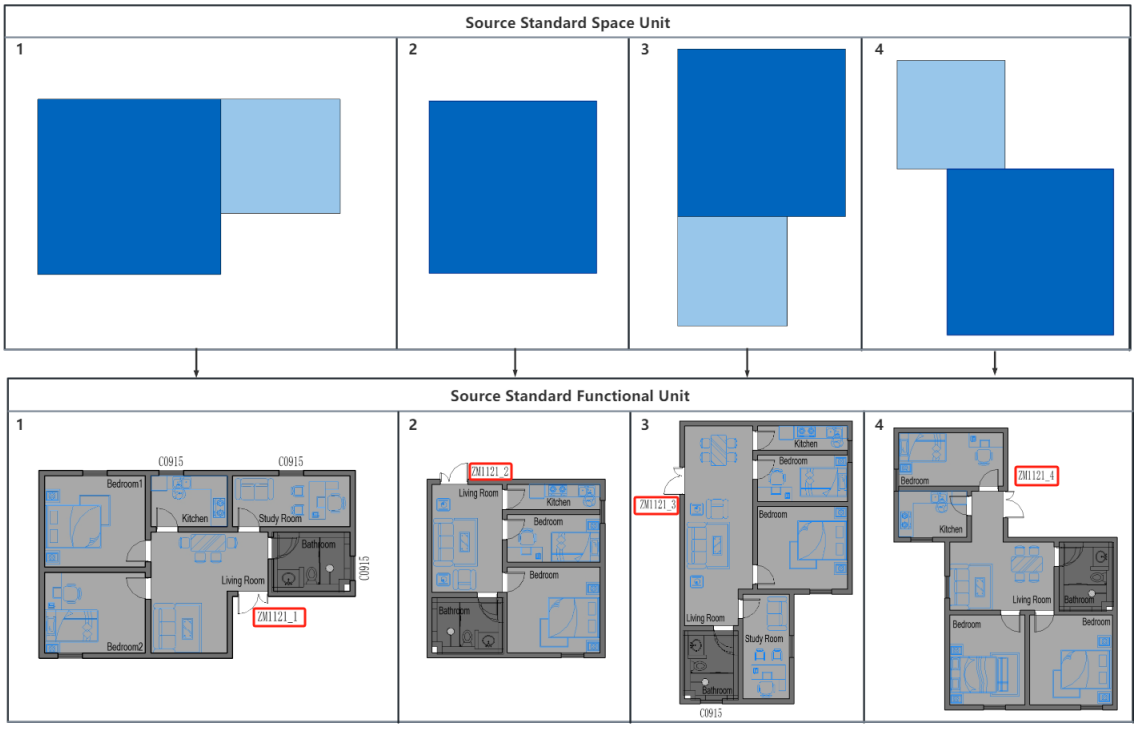

Figure 5.5 Configuration of all source standard functional units

In Figure 5.5, 4 source standard space units correspond to 4 source functional space units. Specifically, in Figure 5.5, the 1st source standard space unit and the 3rd source standard space unit are from the Unit 1 type and its variants illustrated in Figure 5.1, the 2nd source standard space unit is from the Unit 2 type illustrated in Figure 5.1, and the 4th source standard space unit is from the Unit 3 type illustrated in Figure 5.1. The corresponding source standard functional units all have a unique door that enables the units to access one of both corridor units. The door is attached to the wall on the contour of the source standard function space. Of note, different units are labelled with different family type names of the door as follows:

- Source Standard Functional Unit 1: ZM1121_1
- Source Standard Functional Unit 2: ZM1121_2
- Source Standard Functional Unit 3: ZM1121_3
- Source Standard Functional Unit 4: ZM1121_4

By giving an objective standard functional unit the same family type of the door as a source standard functional unit, the source standard functional unit is then paired with the objective standard functional unit, which means that the internal layout of the source standard functional unit can be transformed into the objective standard functional unit. Such pairings are performed when drawing walls with the door on the contour of all objective standard functional units.

After finishing the configuration of all units, the preliminary component-level source and objective floor plans are formed, as illustrated in Figure 5.6:

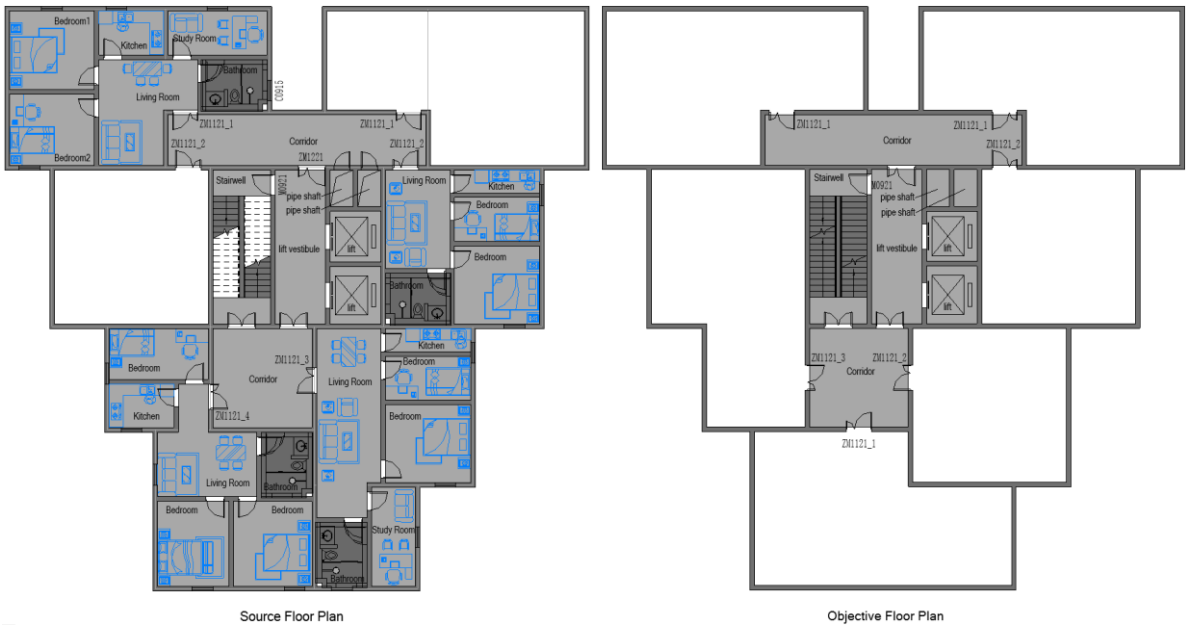

Figure 5.6. Configuration of Preliminary Component-Level Floor Plans

The two floor plans above are preliminary results completed by the designer before entering the automated design process. The preliminary results are categorized into preliminary source floorplans and preliminary objective floorplans. As previously mentioned, the concepts of source and objective floors are relatively contingent on the presence of source standard functional units on the source floor that corresponds to the objective standard functional units on the objective floor.

The tube unit and corridor units have been designed on both source and objective floors. The subjects awaiting transformation are the interior layouts of the source standard functional units, including furniture, assembled rooms, slabs, and walls with attached doors, all situated on the source floor. Transformations are bifurcated into two primary categories: One is the transformation of the internal layout of source standard functional units on the source floor to the matching objective standard functional units on the same floor, and the other is the transformation of the internal layout of source standard functional units on the source floor to the matching objective standard functional units on the objective floor.

All requisite information for these transformations is analyzed and retrieved based on the property graph and the Neo4j graph database, as outlined in Section 5.2. The transformation schemes determining specific transformation types, such as translation, rotation, and mirroring, are generated in Section 5.3.

5.2 Graph Visualization

A Neo4j graph database was initialized using Docker Compose within the PyCharm integrated development environment. Access to the database's web interface, the Neo4j Browser, was facilitated through the URL http://localhost:7474/, as indicated in the PyCharm dashboard. The Neo4j Browser serves as a web-based interface for interaction with the Neo4j database. Authentication to the Neo4j database was achieved using credentials specified in the Docker Compose configuration file. Once authenticated, the Neo4j Browser interface, depicted in Figure 5.7, allowed for the visualization of the graph data. This web-based platform provided the necessary tools for visual and interactive exploration of the graph database.

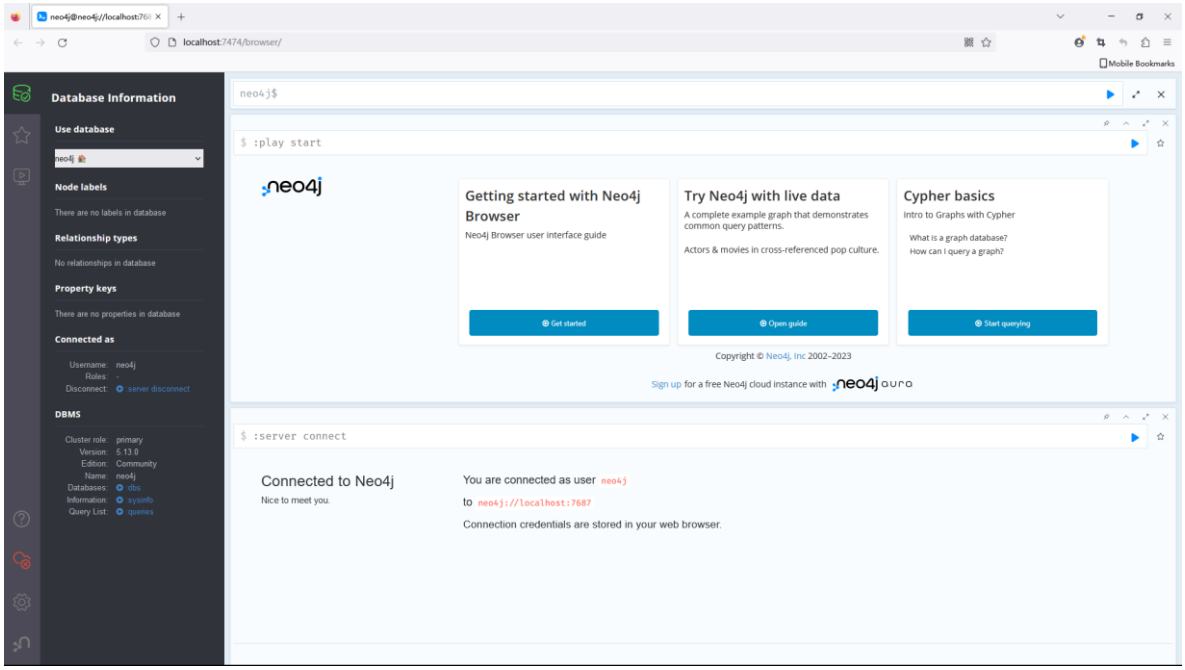

Figure 5.7. Interface of the Created Graph Database in the Neo4j Browser.

Neo4j Python Driver 4.14 is used here to interact with the Neo4j database. Two steps are performed to generate the graph data model in the neo4j database as follows: Firstly, a class is created to encompass methods for establishing and closing a connection to the Neo4j database, then creating, and adding nodes, relationships, and their attributes. In these methods, cypher query statements are embedded. Partial class functions are illustrated as examples in Figure 5.8.

```
01. from neo4j import GraphDatabase
02.
03.
04. class App:
05.
06.     def __init__(self, uri, username, password):
07.         self.driver = GraphDatabase.driver(uri, auth=(username, password))
08.
09.     def close(self):
10.         self.driver.close()
11.
12.     def add_basic_module(self, global_id, components, type_name='basic_module'):
13.         with self.driver.session() as session:
14.             session.execute_write(self.create_basic_node, global_id, components, type_name)
15.
16.     @staticmethod
17.     def create_basic_node(tx, global_id, components, type_name):
18.         tx.run("CREATE (m:Basic_Module "
19.                "{global_id: $global_id, components: $components, "
20.                "type_name: $type_name})",
21.                global_id=global_id, components=components, type_name=type_name)
22.
23.     def add_basic_conjunct_connectivity(self, global_id_a, global_id_b, angle, relative_position):
24.         with self.driver.session() as session:
25.             session.execute_write(self.basic_conjunct_relationship, global_id_a, global_id_b, angle, relative_position)
26.
27.     @staticmethod
28.     def basic_conjunct_relationship(tx, global_id_a, global_id_b, angle, relative_position):
29.         tx.run("MATCH (a:Basic_Module {global_id: $global_id_a})"
30.                "MATCH (b:Conjunctive_Module {global_id: $global_id_b})"
31.                "MERGE (a)-[:CONNECTS {angle: $angle, relative_position: $relative_position}]-(b)",
32.                global_id_a=global_id_a, global_id_b=global_id_b, angle=angle, relative_position=relative_position)
```

Figure 5.8. Partial Cod of Class Methods for Interaction with Neo4j Database.

As illustrated in Figure 5.8, the class is initialized with a Neo4j driver to connect to the Neo4j database. The close method is used to close the driver connection. The “create_basic_node” static method represents a transaction function that executes a Cypher query to create a node labelled as “Basic_Module” with three properties: “global_id”, “components”, and “type_name” with a default value of ‘basic_module’. The “add_basic_module” method adds a “Basic_Module” node to the graph database. It uses the transaction function “create_basic_node” within a session to execute the Cypher query. The “basic_conjunct_relationship” static method represents a transaction function that executes a Cypher query. The query matches nodes with specific global_id properties for both “Basic_Module” and “Conjunctive_Module” labels. It then creates a relationship “CONNECTS” between these nodes with properties “angle” and “relative_position”. The method “add_basic_conjunct_connectivity” is used to add connectivity between a node labelled as “Basic_Module” and another labelled as “Conjunctive_Module” in the graph database. It uses a transaction function, “basic_conjunct_relationship”, within a session to execute a Cypher query.

Secondly, all methods for interaction with the Neo4j Database are executed. A partial code for implementing the operation corresponding to the above figure is illustrated in Figure 5.9.

```
# connect to neo4j
scheme = "bolt"
host_name = "localhost"
port = 7687
url = f"{scheme}://{host_name}:{port}"
username = "neo4j"
password = "Xyz0531!!"
app = neo4j_driver.App(url, username, password)

# create module/node on Neo4J
for basic_module in basic_module_data:
    app.add_basic_module(basic_module['IFC_GUID'],
                         a.getComponentsOfModule(basic_module, elevation_1, wall_width_max, wall_width_min))

# create connectivity/edge on Neo4j
for i in range(len(basic_conjunct_module_connectivity)):
    if basic_conjunct_module_connectivity[i][2] == 1:
        app.add_basic_conjunct_connectivity(basic_conjunct_module_connectivity[i][0],
                                            basic_conjunct_module_connectivity[i][1],
                                            basic_conjunct_module_connectivity[i][6],
                                            basic_conjunct_module_connectivity[i][3])
# close the connection
app.close()

```

Figure 5.9. Partial Code of Executing the Methods in Figure 3.2.4

Through the above two steps, the graph data model for this case can be automatically generated. In the Neo4j browser, by executing the cypher statement **match(n) return n**, the graph can be displayed. All nodes and relationships of the graph are illustrated

in Figure 5.10.

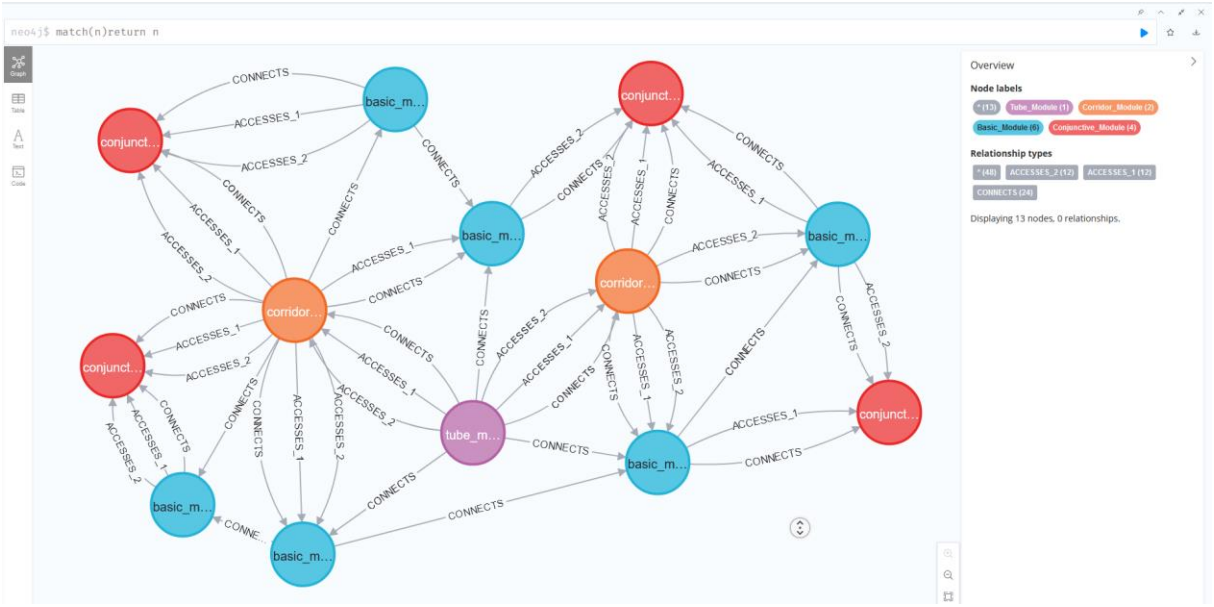

Figure 5.10. Graph Data Model Visualization

5.3 Automated Transformation Implementation

By applying the transformation rules, a file in CSV format can be exported from the PyCharm project that includes a transformation scheme to generate a complete design for the case. The transformation scheme is illustrated in Figure 5.11.

	basic_module_source (1)	source_x (2)	source_...	house_components (4)	basic_module_objective (5)	objective_x (6)	objective_y (7)	floor_name (8)	transform (9)
1	basic_module_source	source_x	source_y	house_components	basic_module_objective	objective_x	objective_y	floor_name	transform
2	0Z3tKe7qv7jOUiPp0aCHLf	-16446	10389	['0i50b5ohf3Kx7suDjP$xGQ', '0i50b5ohf3Kx7suDjP$xHQ', '0i50b5ohf3Kx7suDjP$xHx', '0i50b5ohf3Kx7suDjP$xJM',	0Z3tKe7qv7jOUiPp0aCHKW	1854	10389	House Plan1	Translation + Horizontal Mirror
3	0Z3tKe7qv7jOUiPp0aCHLf	-16446	10389	['0i50b5ohf3Kx7suDjP$xGQ', '0i50b5ohf3Kx7suDjP$xHQ', '0i50b5ohf3Kx7suDjP$xHx', '0i50b5ohf3Kx7suDjP$xJM',	0Z3tKe7qv7jOUiPp0aCHKW	1854	10389	House Plan2	Translation + Horizontal Mirror
4	0Z3tKe7qv7jOUiPp0aCHLf	-16446	10389	['0i50b5ohf3Kx7suDjP$xGQ', '0i50b5ohf3Kx7suDjP$xHQ', '0i50b5ohf3Kx7suDjP$xHx', '0i50b5ohf3Kx7suDjP$xJM',	0Z3tKe7qv7jOUiPp0aCHLf	-16446	10389	House Plan2	Translation
5	0Z3tKe7qv7jOUiPp0aCHLf	-16446	10389	['0i50b5ohf3Kx7suDjP$xGQ', '0i50b5ohf3Kx7suDjP$xHQ', '0i50b5ohf3Kx7suDjP$xHx', '0i50b5ohf3Kx7suDjP$xJM',	0Z3tKe7qv7jOUiPp0aCH4Q	-9996	-7911	House Plan2	Translation + Vertical Mirror
6	0Z3tKe7qv7jOUiPp0aCHtó	-96	3489	['0i50b5ohf3Kx7suDjP$xc1', '0i50b5ohf3Kx7suDjP$xd8', '0i50b5ohf3Kx7suDjP$xdc', '0i50b5ohf3Kx7suDjP$xfP',	250l3GdfrCxOtkrZjNE2D1	-14496	3489	House Plan1	Any
7	0Z3tKe7qv7jOUiPp0aCHtó	-96	3489	['0i50b5ohf3Kx7suDjP$xc1', '0i50b5ohf3Kx7suDjP$xd8', '0i50b5ohf3Kx7suDjP$xdc', '0i50b5ohf3Kx7suDjP$xfP',	0Z3tKe7qv7jOUiPp0aCHtó	-96	3489	House Plan2	Any
8	0Z3tKe7qv7jOUiPp0aCHtó	-96	3489	['0i50b5ohf3Kx7suDjP$xc1', '0i50b5ohf3Kx7suDjP$xd8', '0i50b5ohf3Kx7suDjP$xdc', '0i50b5ohf3Kx7suDjP$xfP',	0Z3tKe7qv7jOUiPp0aCH5b	-3096	-3411	House Plan2	Any
9	0Z3tKe7qv7jOUiPp0aCH5b	-3096	-3411	['0i50b5ohf3Kx7suDjP$xUT', '1DfQwSW2DFoeJN$9VunRR7', '1mvs7WYADB1hT2u$dhH23M', '1mvs7WYADB1hT2u$dhH2BZ',	250l3GdfrCxOtkrZjNE2D1	-14496	3489	House Plan2	Translation + Horizontal Mirror

Figure 5.11 Generated CSV File for Automated Transformation

As illustrated in Figure 5.11, the solution information is as follows:

- IFC_GUID and centre coordinates of the base module in the source standard functional unit

- All the components within the internal layout of the source standard functional unit
- IFC_GUID and centre coordinates of the basic module in the matching objective standard functional unit
- The name of the floor where the matching objective standard functional unit is located.
- The type of transformation from the source standard functional unit to the matching objective standard functional unit.

Since one standard space unit has only one basic module in the case study, the transformation type regarding it is illustrated as "Any" in the CSV file. In this context, the designer needs to manually change the transformation type to a specific type in the CSV file. Otherwise, the transformation is not executed. A feasible change is implemented as follows:

- "Any" of the 6th row in the above figure is changed to "Translation + 270-degree-Counterclockweise Rotation".
- "Any" of the 7th row in the above figure is changed to "Translation".
- "Any" of the 8th row in the above figure is changed to "Translation".

After that, the transformation is implemented in the Revit interface to generate complete source and objective floor plans automatically. The tools used are as follows:

- Specify the IDE Microsoft Visual Studio 2019, where a new class library project is created, C# is selected as the programming language, and .Net Framework 4.8 is specified for compilation. The project is used for CSV file reading and Revit Add-in development.

- Use RevitAPI in the project, as illustrated in Figure 5.12. By using Revit API, a series of interactive operations with Revit can be performed, including translation, rotation, and mirroring between different views.

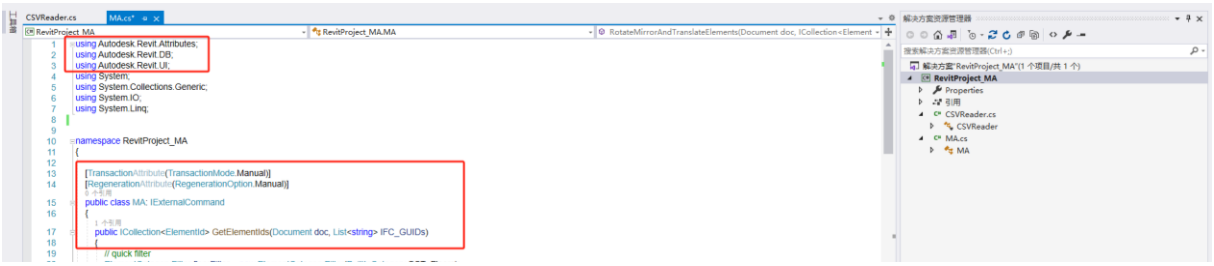

Figure 5.12. Using RevitAPI under the IDE Microsoft Visual Studio 2019

- Two plug-in tools are used to assist in Revit Addin development, as illustrated in Figure 5.13: RevitLookUp and AddInManager. RevitLookUp is used for intuitive and fast access to detailed information about build elements, and AddInManager is used for efficient loading and running of the results of the development.

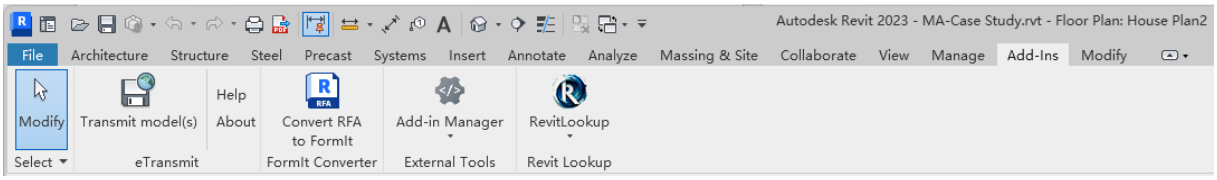

Figure 5.13. Revit plug-in tools: RevitLookUp and AddInManager

After executing the project, a .ddl file is generated. The interface of the project with execution results is illustrated in Figure 5.14.

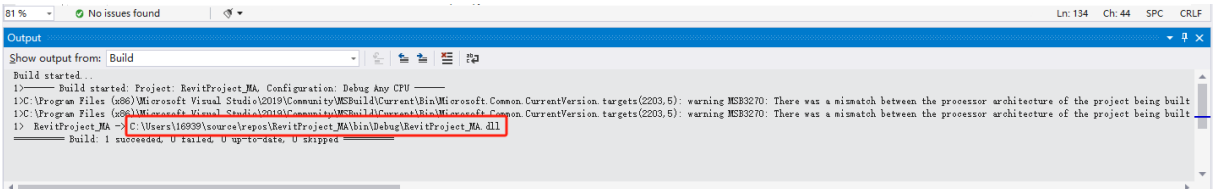

Figure 5.14. .ddl File Generation From the Visual Studio Project

By using Revit Add-in Manager, the generated Addin from the coding project can be loaded into Revit as illustrated in Figure 5.15.

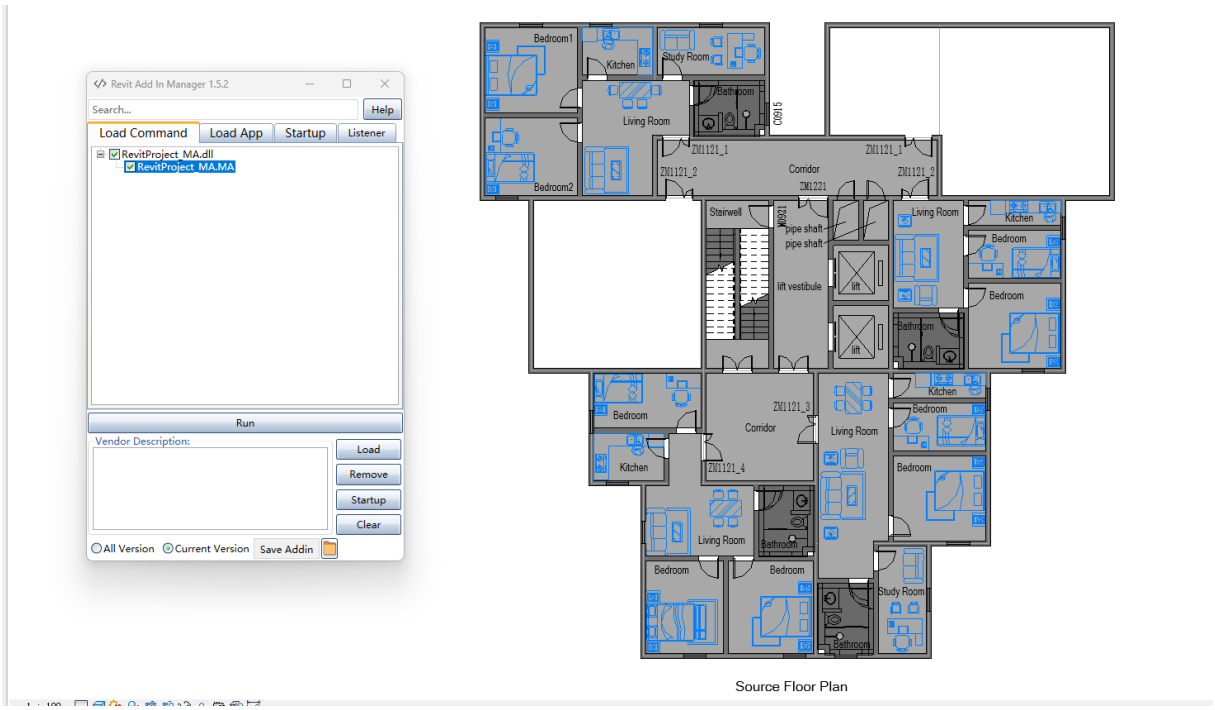

Figure 5.15. .ddl File Loading into Revit

By clicking “run” button illustrated in Figure 5.16, the Addin can be successfully executed, as illustrated in Figure 5.17.

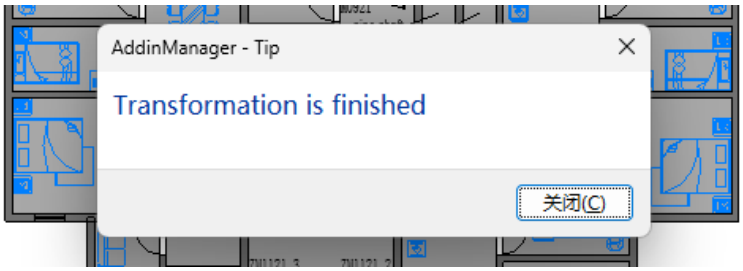

Figure 5.17. Customized Task Dialog at the end of Transformation

The generated compete source and objective floors are illustrated in Figure 5.18.

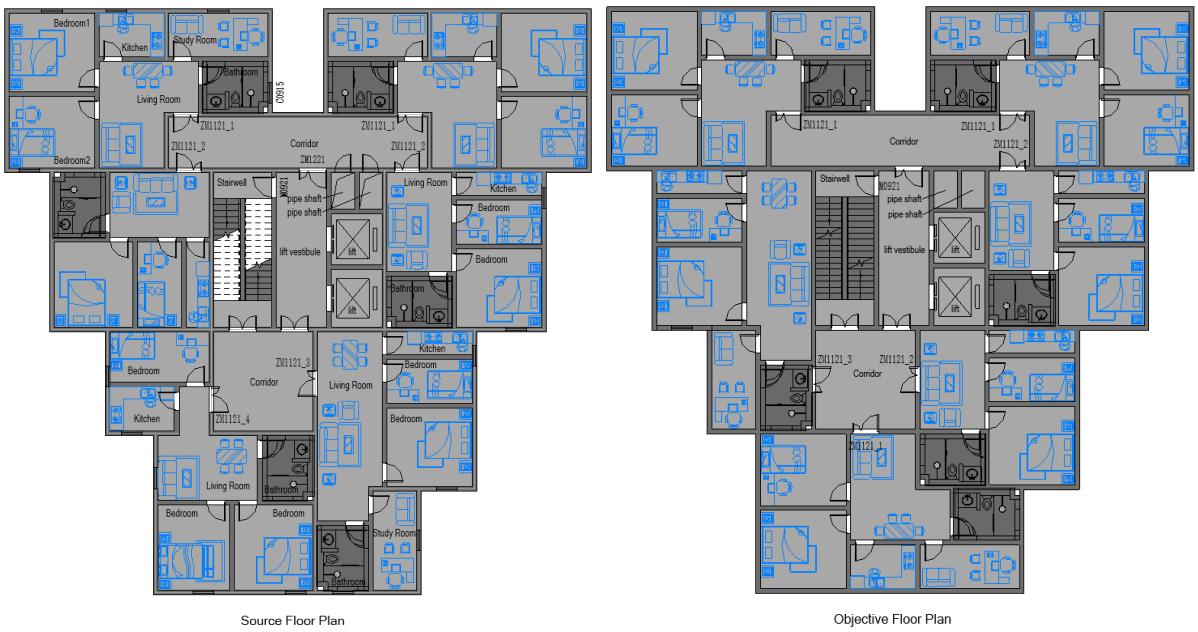

Figure 5.18. Automated Generation of Complete Floorplans

As the transformation scheme illustrated in Figure 5.11, the coordinates of the basic module within a source standard space unit determine the starting point of individual transformation, while the coordinates of the basic module within a matching objective standard space unit determine the ending point of individual transformation. Notably, the transformations of the internal layout of source standard functional units within the source floor are directly executed. However, transformations from the source floor to the objective floor additionally require a copying process. Upon completion of all transformations, a complete source floorplan and a complete objective floorplan are generated.

The programming project "Automated Design Workflow" is hosted on GitLab, accessible via the link https://gitlab.com/Yingz123/automated-design-workflow.git, and the programming project "Revit-Addin for Automated Transformation" is hosted on GitLab, accessible via the link https://gitlab.com/Yingz123/revit-addin-for-automated-transformation.git.

6 Discussion

6.1 Conclusion

One of the key findings of this study is the pivotal role of digital technologies, particularly BIM, in enhancing the standardization and efficiency of the design phase in prefabricated construction. The implementation of BIM has been illustrated to be instrumental in streamlining the design process, allowing for greater flexibility, accuracy, and resource optimization.

The thesis has demonstrated the effectiveness of integrating modularization into the BIM-based architectural design process, thereby addressing a vital need within the AEC industry. It has highlighted the benefits of modularization in its ability to promote the reuse of prefabricated components and prefabricated volumetric modules. In addition, the thesis has also established the feasibility of an automated design workflow, which contributes to the automated design process with IFC file extraction, the graph data model, and graph database.

This thesis provides valuable insights and practical solutions to some of the most pressing challenges in prefabricated construction. It establishes a foundation for future research and development in this field, paving the way for more efficient, sustainable, and cost-effective construction practices. The implications of this research are far-reaching, offering a roadmap for the AEC industry to evolve and adapt in an era increasingly defined by digital innovation and environmental consciousness.

6.2 Limitation

While this thesis has provided valuable insights into the architectural design process for prefabricated buildings, several limitations must be acknowledged to contextualize its findings and implications.

Scope of Application: The research primarily focuses on the design and construction of prefabricated frame-core tube structure buildings. Consequently, the findings and methodologies developed may not be directly applicable or fully transferable to other types of construction projects, such as low-rise residential buildings or specialized architectural structures.

Dependency on Digital Technologies: The proposed design process heavily relies on advanced digital technologies, particularly Building Information Modeling (BIM) and graph databases. This reliance presupposes a certain level of technological infrastructure and expertise, which may not be readily available in all construction contexts, especially in regions with limited technological advancement.

Complexity of Implementation: The implementation of an automated design workflow, while advantageous, introduces a layer of complexity in terms of software proficiency, data management, and integration with existing construction practices. This complexity could pose challenges, particularly for firms or professionals unfamiliar with these technologies.

Empirical Validation: The research would benefit from more extensive empirical testing and real-world case studies. While theoretical and initial practical applications have been explored, broader implementation and testing would provide deeper insights into the practicality and scalability of the proposed design process.

Consideration of External Factors: Factors such as regulatory environments, market dynamics, and socio-economic conditions, which can significantly influence the adoption and success of prefabricated construction methods, have not been extensively explored in this thesis. These external factors play a crucial role in the practical application of the research findings.

Sustainability Considerations: Although the study touches on the potential environmental benefits of modular design, a detailed analysis of the sustainability aspects, including life cycle assessment and carbon footprint analysis, has not been conducted. Such an analysis is critical to fully understand the environmental impact of the proposed design process.

By acknowledging these limitations, the research presents a transparent overview of its scope and applicability. Addressing these limitations in future studies would further enhance the understanding and efficacy of the modularized design process in prefabricated construction.

6.3 Outlook

This thesis sets a foundation for future exploration and innovation in the field of prefabricated design and construction. The research has opened several avenues for

further study and development, which are crucial for advancing the industry. The following outlines the potential future directions:

Integration with emerging technologies: Exploring the integration of newer digital technologies, such as artificial intelligence, machine learning, and advanced data analytics, with the current BIM-based design process could lead to even more sophisticated and efficient design solutions. This could further automate aspects of the design process and enable more personalized and optimized construction outcomes.

Sustainability and environmental impact assessment: Future research should also focus on a comprehensive sustainability assessment of the modularized design process. This includes evaluating the environmental impact, resource efficiency, and life cycle analysis of buildings designed using this approach.

Economic analysis and market viability: Analyzing the economic aspects, such as cost-effectiveness, return on investment, and market viability, is essential. Future studies could explore the economic implications of adopting the approach, providing a more complete picture of its feasibility and attractiveness to stakeholders.

Bibliography

Aldegeily, M. (2018). From architectural design to structural analysis: a data-driven approach to study building information modeling (BIM) interoperability.

Angles, R. (2012, April). A comparison of current graph database models. In *2012 IEEE 28th International Conference on Data Engineering Workshops* (pp. 171-177). IEEE.

Araz; Nasirian; Mehrdad; Arashpour; Babak; & Abbasi, et al. (2019). Optimal work assignment to multiskilled resources in prefabricated construction. *Journal of Construction Engineering and Management*, 145(4).

Barzegar, M., Rajabifard, A., Kalantari, M., & Atazadeh, B. (2021). An IFC-based database schema for mapping BIM data into a 3D spatially enabled land administration database. *International journal of digital earth*, *14*(6), 736-765.

Bayoumi, A. (1999). DESIGN FOR MANUFACTURE AND ASSEMBLY (DFMA): CONCEPTS, BENEFITS AND APPLICATIONS. *Current Advances in Mechanical Design and Production VII*, 501-509.

Beach, T. H., Rezgui, Y., Li, H., & Kasim, T. (2015). A rule-based semantic approach for automated regulatory compliance in the construction sector. *Expert Systems with Applications*, *42*(12), 5219-5231.

Beetz, J., van Leeuwen, J. P., & de Vries, B. (2005). An ontology web language notation of the industry foundation classes. In *Proceedings of the 22nd CIB W78 Conference on Information Technology in Construction* (pp. 193-198). Technische Universität Dresden.

Bian, J., Cao, W., Qiao, Q., & Zhang, J. (2022). Experimental and numerical studies of prefabricated structures using CFST frame and composite wall. *Journal of Building Engineering*, *48*, 103886.

Cao, J., Bucher, D. F., Hall, D. M., & Eggers, M. (2022). A graph-based approach for module library development in industrialized construction. *Computers in Industry*, 139, 103659.

Casini, M. (2016). *Smart buildings: Advanced materials and nanotechnology to improve energy-efficiency and environmental performance*. Woodhead

Publishing.

Chen, C., Tang, L. C. M., & Jin, Y. (2019). Development of 5D BIM-based management system for prefabricated construction in China. In *International Conference on Smart Infrastructure and Construction 2019 (ICSIC) Driving data-informed decision-making* (pp. 215-224). ICE Publishing.

Chen, N., Kan, F., Wang, X., Wang, C., Qi, N., Liu, X., ... & Wang, B. (2016, January). Research on the Secondary Development of Revit Software. In *International Conference on Education, Management, Computer and Society* (pp. 1130-1133). Atlantis Press.

Chen, Z., Pu, Y., & Shelden, D. R. (2019). A Graph Database and Query Approach to IFC Data Management. *Future Inf. Exch. Interoperability*, 28-36.

Chudley, R., & Greeno, R. (2006). *Building construction handbook*. Routledge.

Cui, Y., Li, S., Liu, C., & Sun, N. (2020). Creation and diversified applications of plane module libraries for prefabricated houses based on BIM. *Sustainability*, *12*(2), 453.

Darko, A., Chan, A. P., Yang, Y., & Tetteh, M. O. (2020). Building information modeling (BIM)-based modular integrated construction risk management–Critical survey and future needs. *Computers in Industry*, *123*, 103327.

Divin, N. V. (2020). BIM by using Revit API and Dynamo. A review. *AlfaBuild*, (2), 1404-1404.

Dong, D., Zou, Y., Pan, H., Zhou, G., Feng, Y., & Tang, Y. (2023). DFMA-oriented modular and parametric design and secondary splitting of vertical PC components. *Scientific Reports*, *13*(1), 3457.

Eastman, C. M. (2011). *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons.

Edenhofer, S., Rädler, S., Hoß, M., & Von Mammen, S. (2016, September). Self-organised construction with Revit. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)* (pp. 160-161). IEEE.

Fan, T. W. (2020). Applying fire simulation to BIM modeling with API programming for evacuation time calculation. In *ICACE 2019: Selected Articles from the International Conference on Architecture and Civil Engineering* (pp. 175-184).

Springer Singapore.

Fei, Y., Liao, W., Huang, Y., & Lu, X. (2022). Knowledge-enhanced generative adversarial networks for schematic design of framed tube structures. *Automation in Construction*, *144*, 104619.

Foote, K.D. (2022). Property Graphs vs. Knowledge Graphs. *DATAVERSITY*. https://www.dataversity.net/property-graphs-vs-knowledge-graphs/

Gao, S., Jin, R., & Lu, W. (2020). Design for manufacture and assembly in construction: a review. *Building research & information*, *48*(5), 538-550.

Gibb, A., & Isack, F. (2003). Re-engineering through pre-assembly: client expectations and drivers. *Building research & information*, *31*(2), 146-160.

Gradišar, L., & Dolenc, M. (2021). IFC and Monitoring Database System Based on Graph Data Models. *Advances in Civil Engineering*, *2021*, 1-9.

Hamid, Z., Kamar, K. A. M., Zain, M., Ghani, K., & Rahim, A. H. A. (2008). Industrialized Building System (IBS) in Malaysia: the current state and R&D initiatives. *Malaysia construction research journal*, *2*(1), 1-13.

He, J., Zhou, X., Xu, F., Shi, Y., & Okazaki, T. (2023). Seismic performance of self-centring connections with two energy dissipation stages for reusable modular steel buildings. *Thin-Walled Structures*, 111442.

He, R., Li, M., Gan, V. J., & Ma, J. (2021). BIM-enabled computerized design and digital fabrication of industrialized buildings: A case study. *Journal of Cleaner Production*, *278*, 123505.

Heger, F., & Napps, D. (2023). Floorplan2IFC: Transformation of building floor plans into hierarchical IFC-graphs. *34. ForumBauinformatik*.

Hillier, B., & Hanson, J. (1989). *The social logic of space*. Cambridge university press.

Huahai, Z.; Yuanqi, L. I.; & Yun, L. (2019). A discussion on industrialized residential building design based on standardized and serialized components. *Progress in Steel Building Structures*, 21(03).

Hwang, B. G.; Shan, M.; & Looi, K. Y. (2018). Key constraints and mitigation strategies for prefabricated prefinished volumetric construction. *Journal of Cleaner Production*, 183, 183-193.

Jaillon, L., & Poon, C. S. (2009). The evolution of prefabricated residential building

systems in Hong Kong: A review of the public and the private sector. *Automation in construction*, *18*(3), 239-248.

Jaillon, L., & Poon, C. S. (2014). Life cycle design and prefabrication in buildings: A review and case studies in Hong Kong. *Automation in Construction*, *39*, 195-202.

Janssen, P. (2006). A generative evolutionary design method. *Digital Creativity*, *17*(01), 49-63.

Jin, Z., & Gambatese, J. (2019, June). BIM for temporary structures: Development of a Revit API plug-in for concrete formwork. In *Proceedings of the CSCE Annual Conference Growing with Youth, Laval, QC, Canada* (pp. 12-15).

Kalay, Y. E. (2004). *Architecture's new media: Principles, theories, and methods of computer-aided design*. MIT press.

Kamali, M.; & Hewage, K. (2017). Development of performance criteria for sustainability evaluation of modular versus conventional construction methods. *Journal of cleaner production*, 142, 3592-3606.

Kamar, A. M., Abd Hamid, Z., & Azman, N. A. (2011). Industrialized building system (IBS): Revisiting issues of definition and classification. *International journal of emerging sciences*, *1*(2), 120-132.

Kepczynska-Walczak, A. (2016). 'Building Information Modelling-the Quest for Simplicity Within Complexity'. In *Proceedings of the 34th eCAADe Conference* (Vol. 1, pp. 299-308).

Khalili, A., & Chua, D. H. (2015). IFC-based graph data model for topological queries on building elements. *Journal of Computing in Civil Engineering*, *29*(3), 04014046.

Kim, H. J., Choi, M. H., & Kim, J. J. (2018). A study on the automation process of BIM library creation of air handling unit-development of Revit API module for efficiency and uniformity of library creation. *Journal of the Architectural Institute of Korea Structure & Construction*, *34*(4), 75-82.

Kim, J. H., & Park, I. M. (2013). The practical application of modular construction for residential facilities. *Journal of the Korean housing association*, *24*(3), 19-26.

Kim, M.; Chen, J. C.P.; Sohn, H.; & Chang, C. (2015). A Framework for dimensional

and surface quality assessment of precast concrete elements using BIM and 3D laser scanning. *Automated Construction*, 49, 225-238.

Kovacic, I.; & Zoller, V. (2015). Building life cycle optimization tools for early design phases. *Energy*, 92, 409-419..

Laakso, M., & Kiviniemi, A. O. (2012). The IFC standard: A review of history, development, and standardization, information technology. ITcon, 17(9).

Laovisutthichai, V., Lu, W., & Xue, F. (2021). Modular construction: Design considerations and opportunities. In *Proceedings of the 25th International Symposium on Advancement of Construction Management and Real Estate* (pp. 1351-1361). Springer Singapore.

Li, Y., Gao, Y., Meng, X., Liu, X., & Feng, Y. (2023). Assessing the air pollution abatement effect of prefabricated buildings in China. *Environmental Research*, 239, 117290.

Li, Z.; Shen, G. Q.; & Xue, X. (2014). Critical review of the research on the management of prefabricated construction. *Habitat International*, 43, 240-249.

Li, W., Wan, T., Wang, X., Che, X., Liu, H., & Zhou, Y. (2023). Research on the Application of Mixed Prefabricated Frames in the Seismic Design of Tall Buildings. *Journal of Civil Engineering and Urban Planning*, *5*(5), 46-51.

Liu, H., Singh, G., Lu, M., Bouferguene, A., & Al-Hussein, M. (2018). BIM-based automated design and planning for boarding of light-frame residential buildings. *Automation in Construction*, 89, 235-249.

Lu, W., Tan, T., Xu, J., Wang, J., Chen, K., Gao, S., & Xue, F. (2021). Design for manufacture and assembly (DfMA) in construction: The old and the new. *Architectural Engineering and Design Management*, *17*(1-2), 77-91.

Ma, CY; van Ameijde, J. (2022). Adaptable modular construction systems and multi-objective optimization strategies for mass-customized housing: A new user-driven paradigm for high-rise living in Hong Kong. *International Journal of Architectural Computing*, 20(1), 96-113.

Ma, Z., & Liu, Z. (2018). Ontology-and freeware-based platform for rapid development of BIM applications with reasoning support. *Automation in Construction*, *90*, 1-8.

Moyano, J., León, J., Nieto-Julián, J. E., & Bruno, S. (2021). Semantic interpretation of architectural and archaeological geometries: Point cloud segmentation for HBIM parameterisation. *Automation in Construction*, 130, 103856.

Mulero-Palencia, S., Álvarez-Díaz, S., & Andrés-Chicote, M. (2021). Machine learning for the improvement of deep renovation building projects using as-built BIM models. *Sustainability*, 13(12), 6576.

Nandavar, A., Petzold, F., Nassif, D., Schubert, G., & Ag, B. (2018). Interactive Virtual Reality Tool for BIM Based on IFC. In *Learning, Adapting and Prototyping, Proceedings of the 23rd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)* (pp. 453-462).

Nascimento, D. L. D. M.; Gonalves; & Caiado, R. (2016). Constructability: Modularization and Preassembly using the 3D Model in the Engineering and Procurement Phases for the Construction and Assembly in the Oil and Gas Industry. International Joint Conference - CIO-ICIEOM-IIE-AIM (IJC 2016).

Pan, W., Gibb, A. G., & Dainty, A. R. (2012). Strategies for integrating the use of off-site production technologies in house building. *Journal of construction engineering and management*, *138*(11), 1331-1340.

Pan, Y., & Zhang, L. (2021). Automated process discovery from event logs in BIM construction projects. *Automation in Construction*, 127, 103713.

Panya, D. S., Kim, T., & Choo, S. (2023). An interactive design change methodology using a BIM-based Virtual Reality and Augmented Reality. *Journal of Building Engineering*, *68*, 106030.

Park, J. (2011). BIM-Based Parametric Design Methodology for Modernized Korean Traditional Buildings. *Journal of Asian Architecture and Building Engineering*, 10, 327 - 334.

Parker, H. W. (2017). *Modular for architects* (Doctoral dissertation).

Pauwels, P., & Zhang, S. (2015). Semantic rule-checking for regulation compliance checking: an overview of strategies and approaches. In *32rd international CIB W78 conference*.

Pauwels, P., & Terkaj, W. (2016). EXPRESS to OWL for construction industry:

Towards a recommendable and usable ifcOWL ontology. *Automation in construction*, *63*, 100-133.

Pärn, E. A., & Edwards, D. J. (2017). Conceptualising the FinDD API plug-in: A study of BIM-FM integration. *Automation in Construction*, *80*, 11-21.

Qi, Y.; Chang, S.; Ji, Y.; & Qi, K. (2018). Bim-based incremental cost analysis method of prefabricated buildings in China. *Sustainability*, 10(11).

Ramonell, C., Chacón, R., & Posada, H. (2023). Knowledge graph-based data integration system for digital twins of built assets. *Automation in Construction*, *156*, 105109.

Rausch, C., Edwards, C., & Haas, C. (2020). Benchmarking and improving dimensional quality on modular construction projects–A case study. *International Journal of Industrialized Construction*, *1*(1), 2-21.

Robinson, I.; Webber, J.; & Eifrem, E. (2015). Graph Databases: New Opportunities for Connected Data. O'Reilly Media, Inc.

Saad, A., Ajayi, S. O., & Alaka, H. A. (2023). Trends in BIM-based plugins development for construction activities: a systematic review. *International Journal of Construction Management*, *23*(16), 2756-2768.

Saad, M.; Zhang, Y.; Tian, J.; & Jia, J. (2023). A graph database for life cycle inventory using Neo4j. *Journal of Cleaner Production*, 393.

Sengupta, S., Kanjilal, A., & Bhattacharya, S. (2011). Measuring complexity of component-based architecture: a graph-based approach. *ACM SIGSOFT Software Engineering Notes*, *36*(1), 1-10.

Sarkisian, M. (2016). *Designing tall buildings: Structure as architecture*. Routledge.

Solihin, W., & Eastman, C. (2015). Classification of rules for automated BIM rule checking development. *Automation in construction*, *53*, 69-82.

Sompolgrunk, A., Banihashemi, S., & Mohandes, S. R. (2023). Building information modelling (BIM) and the return on investment: a systematic analysis. *Construction Innovation*, *23*(1), 129-154.

Su, T., Li, H., & An, Y. (2021). A BIM and machine learning integration framework for automated property valuation. *Journal of Building Engineering*, *44*, 102636.

Tak, A. N.; Taghaddos, H.; Mousaei, A.; & Hermann, U. R. (2020). Evaluating industrial

modularization strategies: local vs. overseas fabrication. *Automation in Construction*, 114, 103175.

Tam, C. M., Zeng, S. X., & Deng, Z. M. (2004). Identifying elements of poor construction safety management in China. *Safety science*, *42*(7), 569-586.

Tang, S., & Shelden, D. R. (2020, March). A Framework Utilizing Modern Data Models with IFC for Building Automation System Applications. In *Construction Research Congress 2020* (pp. 11-19). Reston, VA: American Society of Civil Engineers.

Tanyer, A. M., & Aouad, G. (2005). Moving beyond the fourth dimension with an IFC-based single project database. *Automation in Construction*, *14*(1), 15-32.

Tatum, C. B., Vanegas, J. A., & Williams, J. M. (1987). *Constructability improvement using prefabrication, preassembly, and modularization*. Austin, TX: Bureau of Engineering Research, University of Texas at Austin.

Taranath, B. S. (2016). *Structural analysis and design of tall buildings: Steel and composite construction*. CRC press.

Turner, A. (2007). From axial to road-centre lines: a new representation for space syntax and a new model of route choice for transport network analysis. *Environment and Planning B: planning and Design*, *34*(3), 539-555.

Ullah, K., Lill, I., & Witt, E. (2019, May). An overview of BIM adoption in the construction industry: Benefits and barriers. In *10th Nordic conference on construction economics and organization* (pp. 297-303). Emerald Publishing Limited.

Underwood, J., & Isikdag, U. (Eds.). (2009). *Handbook of research on building information modeling and construction informatics: Concepts and technologies: Concepts and technologies*. IGI Global.

Van Landuyt, D., Wijshoff, V., & Joosen, W. (2023). A study of NoSQL query injection in Neo4j. *Computers & Security*, 103590.

Wang, D., & Hu, Y. (2022). Research on the Intelligent Construction of the Rebar Project Based on BIM. *Applied Sciences*, *12*(11), 5596.

Wang, Q., Guo, Z., Mei, T., Li, Q., & Li, P. (2018). Labor crew workspace analysis for prefabricated assemblies' installation: A 4D-BIM-based approach. *Engineering, Construction and Architectural Management*, *25*(3), 374-411.

Wuni, I. Y.; & Shen, G. Q. (2019). Towards a decision support for modular integrated construction: an integrative review of the primary decision-making actors. *International Journal of Construction Management* (1).

Xiao, Y., & Bhola, J. (2022). Design and optimization of prefabricated building system based on BIM technology. *International Journal of System Assurance Engineering and Management*, *13*(Suppl 1), 111-120.

Xu, Z., Zayed, T., & Niu, Y. (2020). Comparative analysis of modular construction practices in mainland China, Hong Kong and Singapore. *Journal of Cleaner Production*, *245*, 118861.

Yeang, K. (2006). Ecodesign: A manual for ecological design. *Wiley.*

Zhang, F., Chan, A. P., Darko, A., Chen, Z., & Li, D. (2022). Integrated applications of building information modeling and artificial intelligence techniques in the AEC/FM industry. *Automation in Construction*, *139*, 104289.

Zhang, Q., Su, Q., & Yan, Y. (2022). Research on Calculating Quantity of Utility Tunnel with Revit Secondary Development. In *Advances in Transportation Geotechnics IV: Proceedings of the 4th International Conference on Transportation Geotechnics Volume 3* (pp. 63-73). Springer International Publishing.

Zhang, X., Zhang, J., Gong, X., & Zhang, S. (2018). Seismic performance of prefabricated high-strength concrete tube column–steel beam joints. *Advances in Structural Engineering*, *21*(5), 658-674.

Zhao, Q., Li, Y., Hei, X., & Yang, M. (2020). A graph-based method for IFC data merging. *Advances in Civil Engineering*, *2020*.

Zhu, J., Wu, P., & Lei, X. (2023). IFC-graph for facilitating building information access and query. *Automation in Construction*, 148, 104778.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelor-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 29. November 2023 [redacted]

Vorname Nachname

[redacted]

Stiftsbogen 64

81375 München

[redacted]