Joint CSCE Construction Specialty & CRC Conference 2025
Conférence conjointe spécialisée en construction de la SCGC et CRC-2025

Montreal, Quebec
July 28-31, 2025 / *28-31 juillet 2025*

GRAPH-BASED GENERATIVE MODEL FOR MULTI-SYSTEM DESIGN SYNTHESIS OF INDUSTRIALIZED HOMEBUILDING

H. Said[1*] and M. Ketabforoush[1]

[1] Civil, Environmental, and Sustainable Engineering, Santa Clara University, CA, USA

ABSTRACT: Industrialized homebuilding is emerging as a key strategy to address the housing crisis by leveraging manufacturing-inspired methods, automation, and material innovation to improve construction speed, quality, and sustainability. Generative design is an essential computational tool in this domain, enabling the automated design of prefabricated housing through the integration of artificial intelligence, parametric modeling, and modularization. However, current generative approaches are constrained by their limited scope—often focusing on individual building systems, offering restricted design variety, and lacking fabrication awareness. This paper presents a novel graph-based generative model for multi-system design synthesis in industrialized homebuilding. The model takes functional requirements and spatial inputs as graphs and applies graph grammar rules to generate a conceptual design graph that integrates key building systems, including structural, mechanical, electrical, and plumbing. The paper outlines the graph metamodel schema, rule sets, and exploration strategies used in the synthesis process. A proof-of-concept implementation is demonstrated through the generative design of an accessory dwelling unit (ADU), showcasing the model's capacity to produce coordinated, fabrication-aware designs from high-level specifications. This work contributes to the literature by expanding the design scope of generative systems to include multi-system integration and supporting early-stage design automation for industrialized construction.

1. INTRODUCTION

Industrialized construction (IC) represents a paradigm shift in the construction industry, characterized by the integration of advanced manufacturing technologies, modularization, and lean principles to enhance efficiency, sustainability, and productivity. Unlike traditional site-intensive construction, IC emphasizes off-site prefabrication of components in controlled environments, enabling improved quality control, reduced construction timelines, and cost efficiencies (Razkenari and Kibert 2022; Yang and Lu 2024). This has been driven by technological advances in areas such as Building Information Modeling, automation, and Industry 4.0 innovations, enabling integrated design, real-time data management, and supply chain logistics optimization (Hadi et al. 2023). Besides, Modular Integrated Construction (MiC) and Industrialized Building Systems (IBS) are flexible and scalable solutions to the increasing global needs for affordable housing and sustainable infrastructure development (Eriksson et al. 2021; Richard 2005). Industrialized construction can be at the forefront in meeting changing requirements, including sustainability concerns, quality issues, labor scarcity, and demographic shifts as society modernizes with increased urbanization.

Generative design is a computational design approach that employs algorithms and artificial intelligence, thus computer-automating the process of generating, analyzing, and optimizing design solutions within preset parameters. Unlike traditional design methods, which involve iterative processes with most of their effort being manual, generative design will explore an enormous range of possibilities by altering the design variables of geometry, material properties, structural performance, and cost constraints (Monizza et al. 2017; Wei et al. 2021). It is a feedback-driven process in which design options are tested based on performance criteria; designers then pick the best that is most efficient, sustainable, and cost-effective (Aksamija et al. 2010). It enhances creativity, but most importantly, increases efficiency by automating repetitive tasks and enabling fast prototyping of very complex architectural forms. In an industrialized construction context, generative design provides a link between mass production and customization. By integrating with Building Information Modeling (BIM) environments, generative algorithms facilitate the design of modular systems optimized for manufacturability, assembly, and sustainability (Wei et al. 2021).

Despite the advances of the previous research, there remains a critical deficiency in integrating several building systems within generative design approaches for industrialized building. Current approaches are limited to a single-system focus, restricted design variety, and lack of fabrication awareness. In this respect, the paper discusses a preliminary graph-based generative model for the design synthesis of industrialized homebuilding systems. Industrialized homebuilding employs advanced computing, automation, and material innovations that could reduce construction time, improve quality, and minimize waste. This model overcomes the limitations of the previous research by transforming input graphs of functional spaces and requirements into a multi-system conceptual design graph, including structural, mechanical, electrical, and plumbing systems, using graph grammars. The specific objectives are: (1) to define a graph metamodel that is able to capture space-function relationships and compose system entities; (2) to induce graph rewriting rules that translate design intent to system-level designs; and (3) to verify the model via a case study of an accessory dwelling unit (ADU).

2. BACKGROUND AND LITERATURE REVIEW

Generative design is a computational approach in AEC that uses algorithms, artificial intelligence, and parametric modeling to generate design alternatives within set parameters (Said 2024). It integrates optimization techniques for structural, aesthetic, and functional criteria and is increasingly implemented within BIM for dynamic design creation and evaluation. This approach has been applied in industrialized construction to optimize mass customization, modular layouts, and prefabrication workflows, such as for glued-laminated timber (Monizza et al. 2017) and modular volumetric architecture (Universidade do Vale do Rio dos Sinos et al. 2019). However, current tools are often limited to single-system optimizations, like architectural or structural layouts, without integrating multidisciplinary systems. This fragmentation highlights the need for holistic frameworks that incorporate architectural, structural, and MEP systems (Aksamija et al. 2010). Parametric design tools like Grasshopper and Dynamo enable generative design by automating workflows and providing real-time performance feedback, reducing material waste, and improving manufacturability. While BIM integration enhances decision-making by supporting coordination across systems, full interoperability remains challenging, especially in large-scale, multi-structural projects.

Current generative approaches often lack fabrication awareness, producing designs that may not be feasible to manufacture efficiently (Wei et al. 2021). To address this, recent research explores graph-based generative systems, which use nodes and edges to represent and optimize building system interdependencies, offering a more integrated approach to design (Abrishami et al. 2020; Kolbeck et al. 2022). Graph-based generative models enable dynamic exploration of design configurations, enhancing collaboration across disciplines and promoting system-level optimization (Gan 2022; Para et al. 2021). These models show promise for overcoming current limitations by integrating multiple building systems and improving the synthesis of design, engineering performance, and manufacturing feasibility. While generative design has made significant strides in industrialized construction, challenges remain in multisystem integration and fabrication awareness. Graph-based models provide a promising path forward, helping to bridge the gap between design intent, engineering, and manufacturing.

Graph-based representations are widely used in civil engineering, architecture, and mechanical product modeling, offering a way to model spatial relationships and connectivity within buildings (Cao et al. 2024;

Said et al. 2024). Specialized graph types, such as port graphs, help clarify the interactions between object nodes and connector nodes, which is especially valuable for evaluating spatial configurations in architecture. These models can integrate geometric and semantic information, making them essential for applications that require design logic alongside computational processes. Graphs have also been used to model buildings at various levels of detail, ensuring consistency across engineering products at multiple scales (Vilgertshofer and Borrmann 2017). The flexibility of graph representations, where nodes and edges can represent entities from simple geometries to complex hierarchical structures, enhances adaptability and automation in engineering workflows (Kolbeck et al. 2022).

Graph rewriting, or graph transformation, is a method used in engineering design to manipulate graph structures by adding, removing, or changing nodes and edges based on predefined rules. This approach automates complex design sequences, facilitates the exploration of design alternatives, and formalizes design knowledge. It provides a structured, consistent way to execute transformations while maintaining flexibility in the design process (Kolbeck et al. 2022). By enabling systematic exploration, it supports the creation of innovative and optimized engineering solutions. Recent advancements in graph transformation techniques have enhanced their applicability in engineering design, particularly in modular construction, where design variants can be generated automatically through parametric and generative models based on predefined criteria (Voss et al. 2023). Rule-based transformations help automate design processes, reducing manual effort and improving efficiency. Additionally, the integration of knowledge-based and generative approaches enables predictive modeling, contributing to better decision-making early in the design process (Aksamija et al. 2010). Graph grammars, an extension of graph rewriting, provide a formal framework for generating and manipulating structures. These grammars enable systematic exploration of design spaces, creating valid configurations through structured transformation rules (Campbell 2009). They are particularly effective in computational design synthesis, allowing engineers to encode domain-specific knowledge into transformation rules that can generate design alternatives. By using metamodels or type graphs to define components and their relationships, graph-based design ensures coherence, correctness, and adherence to engineering principles in automated design synthesis (Schmidt and Rudolph 2016). As computational capabilities improve, graph rewriting techniques are expected to play an even larger role in automating and optimizing design processes across multiple engineering disciplines (Ambite et al. n.d.; Kolbeck et al. 2022).

While generative design has made significant strides in industrialized construction, current tools often remain fragmented—focusing on single-system optimizations or lacking fabrication awareness. This reveals a gap in the development of holistic generative frameworks that integrate architectural, structural, and MEP systems while also ensuring constructability. Addressing this gap, the proposed research introduces a unified graph-based model to bridge design intent, multidisciplinary integration, and fabrication feasibility (Aksamija et al. 2010; Wei et al. 2021).

3. PROPOSED GRAPH-BASED GENERATIVE MODEL

The proposed model, called Building Graph-based Generative Synthesis (BG^{2}S) model, utilizes graph modeling and graph rewriting (i.e. transformation) to synthesize the design of a building and the underlying main building systems from the building's space functional programming. The development of BG^{2}S required the definition of three main graph grammar components: the type graph, a starting graph, and the graph rewriting rules. The model was implemented using GROOVE (Ghamarian et al. 2012), an open-source graph modeling and transformation tool that enables easy but robust graph-based representation of system design and process behaviors. In this paper, the BG2S model is described in step-by-step, reproducible way. The four key steps involved in building the model were: (1) developing the building type graph (ontology), (2) inputting the start graph for a project's design requirements, (3) developing of the design synthesis graph rewriting rules, and (4) the simulation of the rule application via GROOVE software.

3.1 Building Type Graph

A type graph is the metamodel of the start graph and the applied graph rewriting rules. The type graph provides a more defined and controlled way to model a system by determining the allowed combinations of node, edges, and attributes. Developing a type graph from scratch is a time consuming and an intellectual intense process. As such, it is recommended to utilize an existing ontology model that is close to the application of the graph rewriting model. For this study, multiple ontologies were considered, including BuildingMOTIF, Project Haystack, GreenBuildingXML, and BRICK. However, RealEstateCore (REC) Ontology was selected due to its comprehensive and generic modeling of a facility without a narrow focus on smart building systems (like Project Haystack or BRICK) or green building (like GreenBuildingXML). The development of REC ontology involved a collaboration with BRICK developers by overlapping the classes used to model the building equipment and automation systems.

The type graph of REC is shown in Figure 1, which is mostly adopted for the development of the proposed BG^{2}S model. REC includes six main super classes: Asset, Space, Building Element, Agent, Collection, and Point. As this study mainly focuses on the design phase of a building, only the former three classes (Asset, Space, Building Element) and most of their subclasses are considered. The latter super classes were not considered because of their relevance to the operation phase of the building. The REC ontology needed to be augmented to achieved the desired outcome of BG^{2}S, which include a new class (Function) and two new relations for the Space class. As such, the main classes of BG^{2}S ontology are:

- An Asset is an object that is placed inside of a building, but is not an integral part of that building's structure. The Asset subclasses include Equipment, Architectural Asset, and Furniture. The Equipment class represents a significant part of the REC ontology and BG^{2}S type graph, as it includes 21 subclasses that represent the main building systems (electrical, fire safety, HVAC, water, etc). The Architectural Asset mainly represent barrier assets, like Door, Partition, and Window. The Furniture asset is not included in BG^{2}S due to its weak influence on the overall synthesis of the building design. An Asset can have parts of sub-assets, can be mounted on a building element, and is located in a specific space. The Equipment class has an additional unique relation to represent the feeding of matter (fluid, power, information) between the equipment.
- A Space is contiguous part of the physical reality that includes constructed spaces (under its subclass Architecture) and other natural spaces. The Architecture space involves seven subclasses that represent hierarchical presentation of the faculty constructed space. For the minimal viable use of BG^{2}S, only *Room* and *OutdoorSpace* can be used. Three new relations are defined for the Space class to allow the representation of schematic design of a building: *isAdjacentTo* and *isConnectedTo*. The first relation is used to represent desired adjacency between two spaces by sharing a wall. On the other hand, the second relation is used to represent a circulation linkage between two spaces that requires the existence of a door. The third relation declares the requirement to achieve a function within the space, as explained later.
- A Building Element is a component of the building the represents a part of the main structural system, such as wall, slab, roof, façade, and balcony. A Builder element can be part of another element, can be related to Architectural spaces using a containment, adjacency, and intersection relations.
- A Function is class that augments the REC ontology to represent a desired function within a space. A Function has a unique ID number, which is based on a standard list of building functions that are commonly used in the space programming process in a building architectural design.

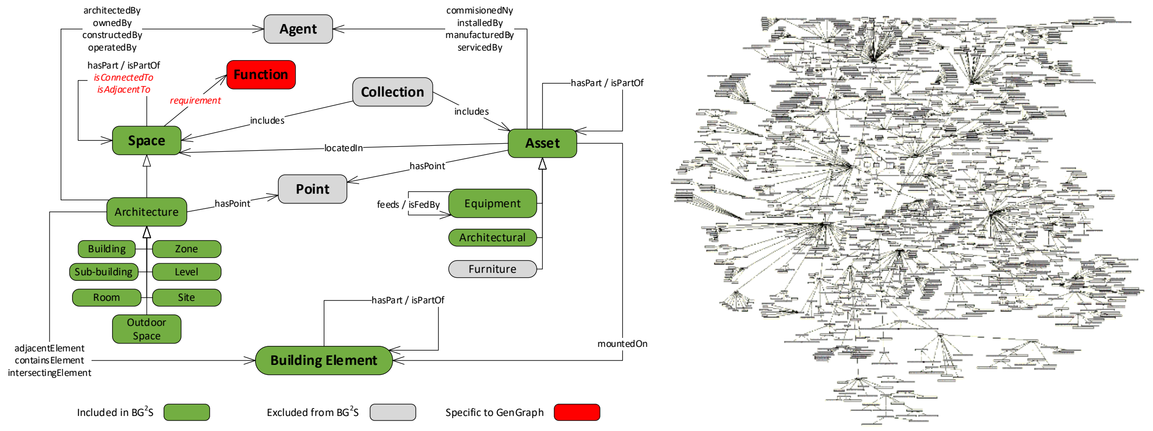

Figure 1: A simplified UML representation of the building type graph utilized in GraphGen (left), and a screenshot of the fully implement ontology in GROOVE (right).

3.2 Starting Graph

The starting graph is a project-specific graph that represents the design intent in the form of a space programming diagram. The starting graph creates instances of the type graph to represent the main building spaces, the functional requirements, the relations between the spaces, and the overall orientation of the building. Figure 2 depicts a start graph of a small ADU building, which includes 7 rooms and 26 functions.

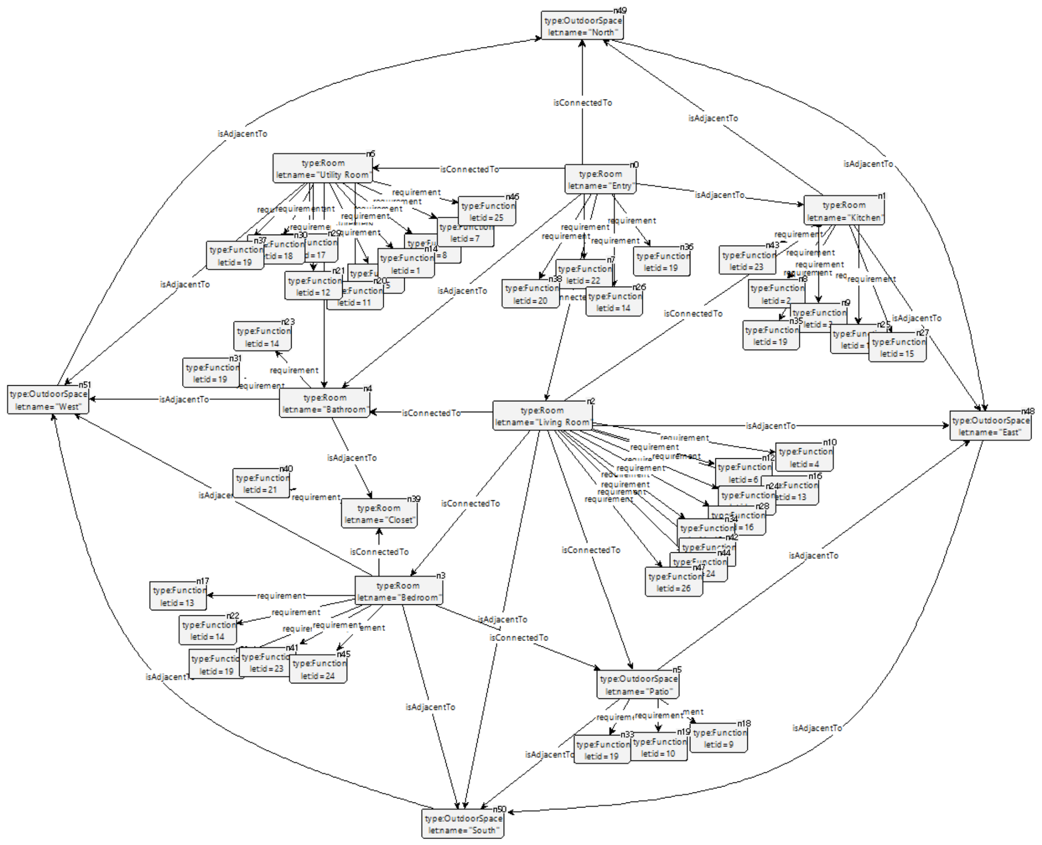
type:OutdoorSpace letname="North"
type:Room letname="Utility Room"
type:Room letname="Entry"
type:Room letname="Kitchen"
type:OutdoorSpace letname="West"
type:Room letname="Bathroom"
type:Room letname="Living Room"
type:OutdoorSpace letname="East"
type:Room letname="Closet"
type:Room letname="Bedroom"
type:OutdoorSpace letname="Patio"
type:OutdoorSpace letname="South"
isConnectedTo
isAdjacentTo
requirement
type:Function

Figure 2: The start graph if a small ADU building.

Each of these rooms is related to other rooms using either “isConnectedTo” or “isAdjacentTo”. For example, the Entry room is connected to two rooms (Living and Utility), and is adjacent to two other rooms (Bathroom and Kitchen). It should be noted that type `OutdoorSpace` is used to define the four geographic orientation directions (North, East, South, West). These outdoor spaces are important to define the orientation of the building and also to differentiate between interior and exterior walls (as explained later). These orienteering outdoor spaces is linked to each other using “*isAdjacentTo*” relations in a circular fashion. In addition, the building rooms can be linked to these orienteering outdoor spaces using either a connection (i.e. an exterior wall with a door) or adjacency relation (i.e. an exterior wall with a window). It’s *worthnoting* that the `OutdoorSpace` class is used to define a functional space outside the building, i.e., a patio space.

Each of the building spaces should be linked to at least one Function node. For the illustrated ADU example, a list of 26 functional requirements was created and assigned to the different rooms. For example, Function number 7 calls for “regulating the air temperature, humidity, and quality” and it is assigned to the Utility Room space. These functional requirements are critical as they represent the seed of the design synthesis (Kurtoglu and Campbell 2006), as explained in the following section.

3.3 Graph Rewriting Rules

The graph transformation is achieved using a set of graph-rewriting rules, which are used in BG^{2}S to translate the start graph of the building space function requirements into a synthesized multi-system design. Graph rewriting (or transformation) rules define how a graph's structure can be modified by specifying patterns for adding, removing, or altering nodes and edges. These rules operate based on a formalized approach where transformations follow predefined conditions to ensure logical consistency. Each rule consists of two key components:

- Left-Hand Side (LHS) – This represents the pattern that must be identified within the graph before a transformation can take place. The LHS specifies a subgraph structure, including specific nodes, edges, and possibly attributes, that must exist in the graph for the rule to be applicable.
- Right-Hand Side (RHS) – This defines the modified structure that replaces the matched pattern. The RHS describes how the identified subgraph changes after applying the rule, which may involve adding new elements, deleting existing ones, or modifying relationships between components.

When a graph rewriting rule is applied, the algorithm searches the graph for occurrences of the LHS pattern. Once a match is found, it replaces that subgraph with the RHS structure according to the transformation rule. This mechanism allows for systematic updates to the graph, ensuring controlled modifications in design and engineering applications. Graph rewriting rules can be applied iteratively, leading to the progressive evolution of a design or system while maintaining logical and structural coherence.

Graph rewriting rules are needed transformed a space-function (S-F) start graph into a synthesized space-element-Asset (S-E-A) graph. As such, five main groups of graph rewriting rules were formulated: function-asset (F-A) rules, asset-asset (A-A) rules, space-element (S-E) rules, element-element (E-E) rules, and element-asset (E-A) rules. The following subsections briefly explains each of these rewriting rule groups, as well as the application of these rule to explore the space and process of synthesizing the design.

3.3.1 Function-Asset (F-A) Rules

The goal of this set of rules is to translate the functions within each room into a set of assets. This resembles the function-mean diagram approach (Robotham 2002) that has been used previously in modeling industrialized modular building (Eriksson et al. 2021). In BG^{2}S context, the mean is replaced with the building assets as building equipment are included in the project model to replace the spatial functions in each room. This approach requires one-to-one mapping of the functions and assets. As such, the ADU illustrative example starts with 26 space functional requirements and will transform into adding 26 building assets. Figure 3 shows two examples of these F-A rules, which look similar due to the adopted function-asset mapping. Each rule looks for the matching of the blue and black nodes of the subgraph in the start graph. Once this matching is found, the blue nodes (the function) is removed and replaced wit the green node (the asset). The depicted left rule handles the replacement of function 21 "Store surveillance footage" with a *Network_Video_Recorder* typed node. The red nodes "*forallx:*" is a GROOVE feature to create nested rules to enable the application of the rule for multiple matchings in the start graph all at once.

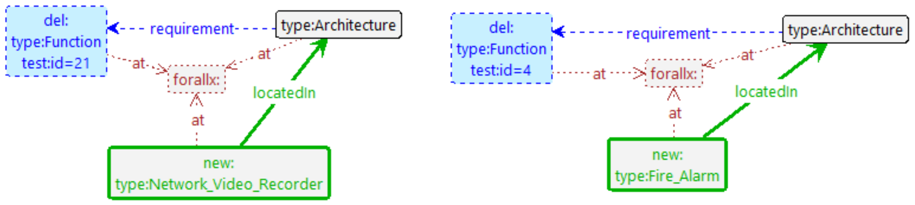

Figure 3: Sample function-asset (F-A) rules: adding a network video recorder (left) and a file alarm (right)

3.3.2 Asset-Asset (A-A) Rules

The A-A rules links between the building equipment using feed relations, following the commonly established logic chains of building systems (Xiao et al. 2019). The rule shown in Figure 4-a detects the existence of the *Surveillance_Camera* and the *Network_Video_Recorder*, and links between them using a conditional "*cnew*" prompt that creates a "feeds" relation only if it has existed already. Figure 4-b shows a similar rule but for a more connected building asset (breaker panel).

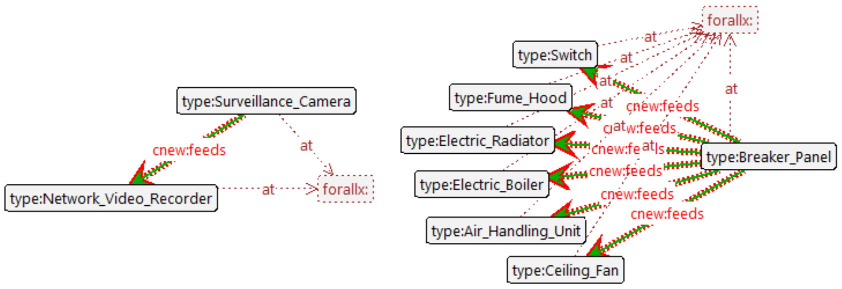

Figure 4: Sample asset-asset (A-A) rules: creating feeds links between surveillance (left) and electrical (right) systems

3.3.3 Space-Element (S-E) Rules

This S-E rule set add the basic building element (structural components) to the model based on the topology of the building spaces. Figure 5-a shows the rule for creating a *WallInner* entity and a *Door* mounted on it between two rooms if they are connected. On the other hand, Figure 5-b shows another rule that inserts an external wall with a mounted window between a room and an adjacent outdoor space. Finally, Figure 5-c shows a simpler rule to add a roof and a slab to any room that does have yet these elements.

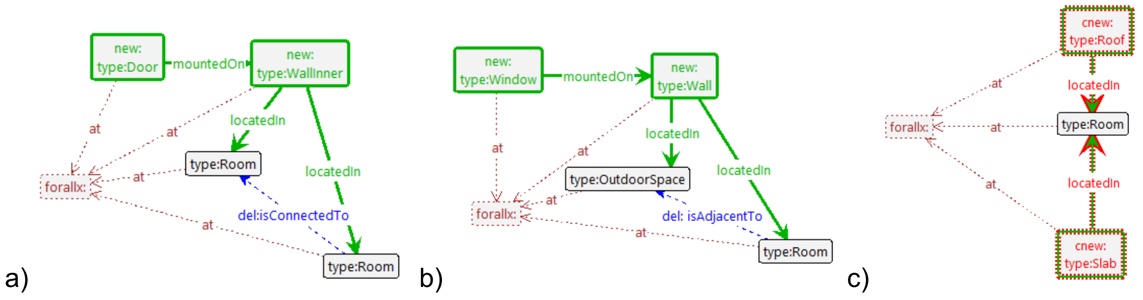

Figure 5: Sample space-element (S-E) rules: creating a) an inner wall with a door, b) an exterior wall with

a window, c) a slab and roof

3.3.4 Element-Element (E-E) Rules

After the elements are added to the building graph, the E-E rules connect between them based on their topology in the building. Figure 6-a shows a sample rule that connects between roof and slab elements if they are located in rooms that share a common wall. Figure 6-b shows a sample rule that connects between three walls that are shared between three architecture spaces (either a room or an outdoor space).

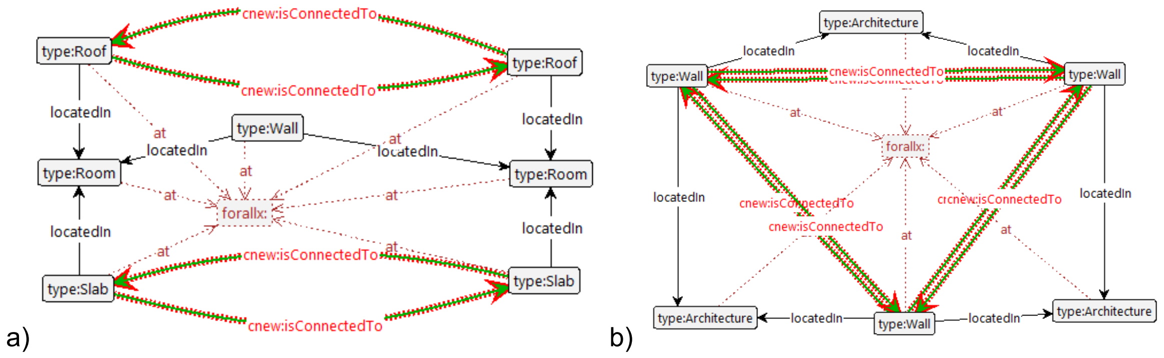

Figure 6: Sample element-element (E-E) rules: connecting between: a) slabs and roofs, 2) walls

3.3.5 Element-Asset (E-A) Rules

The last set of rewriting rules attempt to link between the building assets and elements to reflect their physical dependency. Figure 6-a shows a rule that attempts to mount a humidifier on a slab element in an architecture space if it was not mounted already on a roof. There is a similar rule that attempts the reversed mounting logic, favoring roof mounting over slab mounting. These two rules represent different mounting possibilities, which create different paths in the design synthesis space. Figure 6-b shows another rule that attempts to mount a breaker panel on a wall if it's not already placed on another wall in the same space.

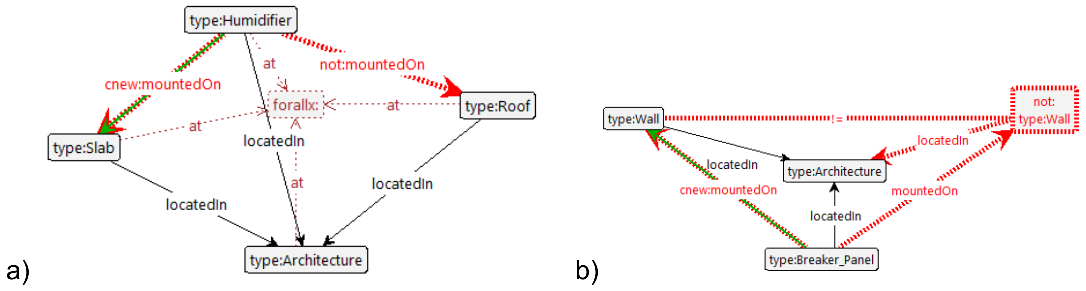

Figure 6: Sample element-asset (E-A) rules: mounting a) humidifier on a slab; b) a breaker panel on a wall

3.3.6 Rule application and design synthesis simulation

GROOVE is used to simulate the application of the design synthesis rules for the small ADU building. The simulation tracks the transformation of the building design graph through synthesis states, where a transition happens between states by applying of the rules that are matched in the latest graph. Figure 7 shows the beginning of this simulation process that is limited to only three transition steps of applying the possible rewriting rules. When this simulation is completed, the graph of the synthesized design is created, as shown in Figure 7 (right photo). It should be noted that some of the E-A rules, related to mounting asset on walls, were disabled in this simulation due to the significant exploration space that was realized from all the possibility of mounting all the assets on all the walls.

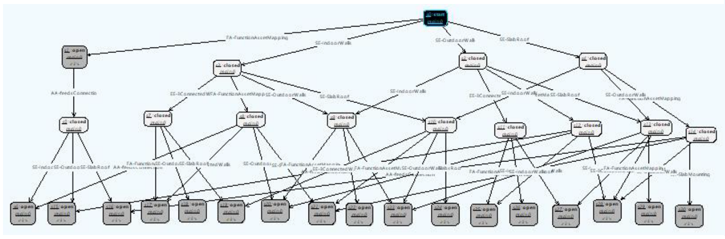
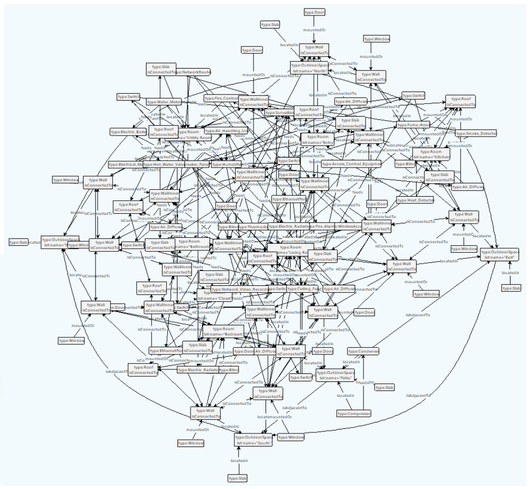

Figure 7: The design synthesis space simulation (left) and the synthesized design graph (right)

4. CONCLUSIONS

This paper presents a graph-based generative model for synthesizing multi-system designs in industrialized homebuilding, aiming to enhance efficiency, quality, and sustainability. By combining AI-driven generative design with modularization and off-site prefabrication, the model addresses key challenges in industrialized construction, including limited design variability and lack of fabrication awareness. It transforms functional space requirements into conceptual design graphs integrating structural, mechanical, electrical, and plumbing systems using graph grammars. Implemented within a BIM environment, the approach supports automated, constructible, and adaptable modular housing solutions. A case study on accessory dwelling units (ADUs) illustrates the model's ability to generate high-performance, fabrication-aware designs aligned with industrialized construction goals.

The study is in its preliminary stages with acknowledged limitations that necessities future research developments and expanded research studies. The BG^{2}S should be applied to a larger project with more complicated functional requirements. Also, the function-asset mapping and rules should be expanded and validated by detailed MEP system literature review, expert interviews, and empirical testing. In addition, these rules have been crafted manually, while future studies can explore the use of natural language processing to automatically detect and draft them from existing design manuals. Finally, a more advanced rewriting model can be developed in a generic scripting environment that would allow the handling of larger start graph and more rewriting rules. Also, this future advanced model can allow the integration of the rewriting methodology with other critical design tools, such as building information modeling and building energy simulation.

REFERENCES

Abrishami, S., J. Goulding, and F. Rahimian. 2020. "Generative BIM workspace for AEC conceptual design automation: prototype development." *Eng. Constr. Archit. Manag.*, 28 (2): 482–509. https://doi.org/10.1108/ECAM-04-2020-0256.

Aksamija, A., K. Yue, H. Kim, F. Grobler, and R. Krishnamurti. 2010. "Integration of knowledge-based and generative systems for building characterization and prediction." *Artif. Intell. Eng. Des. Anal. Manuf.*, 24 (1): 3–16. https://doi.org/10.1017/S0890060409990138.

Ambite, J. L., C. A. Knoblock, and S. Minton. n.d. "Plan Optimization by Plan Rewriting." 41.

Campbell, M. 2009. "A Graph Grammar Methodology for Generative Systems." 25.

Cao, J., H. Said, A. Savov, and D. Hall. 2024. "Graph-Based Evolutionary Search for Optimal Hybrid Modularization of Building Construction Projects." *J. Constr. Eng. Manag.*, 150 (8): 04024098. American Society of Civil Engineers. https://doi.org/10.1061/JCEMD4.COENG-14687.

Eriksson, H., M. Sandberg, G. Jansson, and J. Lessing. 2021. "Exploring Product Modularity in Residential Building Areas." *Buildings*, 11 (7): 281. Multidisciplinary Digital Publishing Institute. https://doi.org/10.3390/buildings11070281.

Gan, V. J. L. 2022. "BIM-based graph data model for automatic generative design of modular buildings." *Autom. Constr.*, 134: 104062. https://doi.org/10.1016/j.autcon.2021.104062.

Ghamarian, A. H., M. de Mol, A. Rensink, E. Zambon, and M. Zimakova. 2012. "Modelling and analysis using GROOVE." *Int. J. Softw. Tools Technol. Transf.*, 14 (1): 15–40. https://doi.org/10.1007/s10009-011-0186-x.

Hadi, A., F. Cheung, S. Adjei, and A. Dulaimi. 2023. "Evaluation of Lean Off-Site Construction Literature through the Lens of Industry 4.0 and 5.0." *J. Constr. Eng. Manag.*, 149 (12): 03123007. American Society of Civil Engineers. https://doi.org/10.1061/JCEMD4.COENG-13622.

Kolbeck, L., S. Vilgertshofer, J. Abualdenien, and A. Borrmann. 2022. "Graph Rewriting Techniques in Engineering Design." *Front. Built Environ.*, 7: 815153. https://doi.org/10.3389/fbuil.2021.815153.

Kurtoglu, T., and M. I. Campbell. 2006. "A GRAPH GRAMMAR BASED FRAMEWORK FOR AUTOMATED CONCEPT GENERATION." *36 Proc. Des. 2006 9th Int. Des. Conf. Dubrov. Croat.*, 61–68.

Monizza, G. P., E. Rauch, and D. T. Matt. 2017. "Parametric and Generative Design Techniques for Mass-Customization in Building Industry: A Case Study for Glued-Laminated Timber." *Procedia CIRP*, 60: 392–397. https://doi.org/10.1016/j.procir.2017.01.051.

Para, W., P. Guerrero, T. Kelly, L. Guibas, and P. Wonka. 2021. "Generative Layout Modeling using Constraint Graphs." *2021 IEEECVF Int. Conf. Comput. Vis. ICCV*, 6670–6680. Montreal, QC, Canada: IEEE.

Razkenari, M., and C. J. Kibert. 2022. "A Framework for Assessing Maturity and Readiness Towards Industrialized Construction." *J. Archit. Eng.*, 28 (2): 04022003. American Society of Civil Engineers. https://doi.org/10.1061/(ASCE)AE.1943-5568.0000528.

Richard, R.-B. 2005. "Industrialised building systems: reproduction before automation and robotics." *Autom. Constr.*, 20th International Symposium on Automation and Robotics in Construction: The Future Site, 14 (4): 442–451. https://doi.org/10.1016/j.autcon.2004.09.009.

Robotham, A. J. 2002. "The use of function/means trees for modelling technical, semantic and business functions." *J. Eng. Des.*, 13 (3): 243–251. https://doi.org/10.1080/09544820110108944.

Said, H. 2024. "Generative Design Effectiveness-Constructability Tradeoff of Panelized Prefabricated Buildings." *2024 ASCE Int. Conf. Comput. Civ. Eng.* Pittsburgh, Pennsylvania, USA: ASCE.

Said, H., A. Rajagopalan, and D. M. Hall. 2024. "Longitudinal analysis of interorganizational collaborative networks of cross-laminated timber (CLT) construction." *Constr. Innov.*, ahead-of-print (ahead-of-print). Emerald Publishing Limited. https://doi.org/10.1108/CI-01-2023-0012.

Schmidt, J., and S. Rudolph. 2016. "Graph-Based Design Languages: A Lingua Franca for Product Design Including Abstract Geometry." *IEEE Comput. Graph. Appl.*, 36 (5): 88–93. https://doi.org/10.1109/MCG.2016.89.

Universidade do Vale do Rio dos Sinos, A. Teribele, B. Turkienicz, and Universidade Federal do Rio Grande do Sul. 2019. "Generative model and fixing guidelines for modular volumetric architecture." *Rev. Constr.*, 17 (3): 517–530. https://doi.org/10.7764/RDLC.17.3.517.

Vilgertshofer, S., and A. Borrmann. 2017. "Using graph rewriting methods for the semi-automatic generation of parametric infrastructure models." *Adv. Eng. Inform.*, 33: 502–515. https://doi.org/10.1016/j.aei.2017.07.003.

Voss, C., F. Petzold, and S. Rudolph. 2023. "Graph transformation in engineering design: an overview of the last decade." *AI EDAM*, 37: e5. Cambridge University Press. https://doi.org/10.1017/S089006042200018X.

Wei, Y., H. Choi, and Z. Lei. 2021. "A generative design approach for modular construction in congested urban areas." *Smart Sustain. Built Environ.* https://doi.org/10.1108/SASBE-04-2021-0068.

Xiao, Y.-Q., S.-W. Li, and Z.-Z. Hu. 2019. "Automatically Generating a MEP Logic Chain from Building Information Models with Identification Rules." *Appl. Sci.*, 9 (11): 2204. Multidisciplinary Digital Publishing Institute. https://doi.org/10.3390/app9112204.

Yang, Z., and W. Lu. 2024. "Lean Modular Integrated Construction Production Phase Planning under Uncertainties: A Big Data–Driven Optimization Approach." *J. Constr. Eng. Manag.*, 150 (6): 04024048. American Society of Civil Engineers. https://doi.org/10.1061/JCEMD4.COENG-14420.