PBG: A parametric building graph capturing and transferring detailing patterns of building models

Jimmy Abualdenien, jimmy.abualdenien@tum.de
Technical University of Munich, Munich, Germany

André Borrmann, andre.borrmann@tum.de
Technical University of Munich, Munich, Germany

Abstract
Design and detailing decisions result from numerous considerations and boundary conditions. Such decisions highly influence the cost and performance of the final design. Typically, architects and engineers tend to employ their domain knowledge and reuse successful detailing patterns that fulfill the current needs and boundary conditions. Detailing patterns are described through building information and the rationale behind them. This paper presents a Parametric Building Graph (PBG) for capturing detailing patterns. Additionally, it proposes a framework for automatically transferring those patterns to new projects. In more detail, detailing patterns are stored as subgraph templates, and then when detailing a new building, a pattern is matched and replaced across a graph representation of the building using Graph Rewriting Systems (GRS). Finally, the detailed building graph is brought back to the BIM-authoring tool. The paper is concluded with a feasibility study that demonstrates the realization of the proposed approach in a prototype and a use case.

Keywords: Graph Representation, Graph Rewriting, Detailing Decisions, Detailing Patterns

1 Introduction

Building designs are wealthy with numerous implicit design decisions and domain knowledge. Every construction project must fulfill various owner requirements, regulations, as well as boundary conditions (Strug & Ślusarczyk 2017). Accordingly, as depicted in Figure 1, satisfying these requirements is reflected within the selected architectural concepts (concept level).

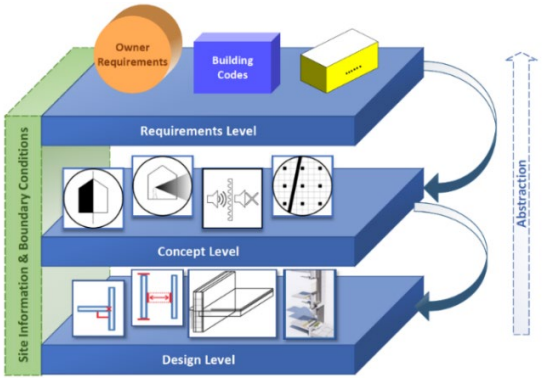

Figure 1. Illustration of the design process through abstraction levels

The selected concepts are then realized through modeling and detailing the individual elements, including their geometric and semantic information as well as their topological

relationships and functional dependencies (design level). For example, a site of a residential building close to a highway (where traffic is high) requires careful consideration of the designed facade, especially in terms of noise reduction techniques. On the other hand, a site facing a nature preserve fosters using curtain walls or big windows. Detailing decisions can be as simple as deciding on the position of a staircase or as complex as selecting the type of junction between walls and slabs, including choosing the combination of their material layers (Schneider-Marin & Abualdenien 2019).

Detailing decisions significantly influence the performance of the resultant building design from various aspects, including energy efficiency, cost, and comfort (Abualdenien & Borrmann 2019). Hence, designers typically produce and detail multiple design options to explore and evaluate several possibilities at the different phases (Exner et al. 2019). Furthermore, although each construction project is unique in its context, designers tend to rely on their domain knowledge gained from previous successful projects, following a similar combination of building information and their dependencies for achieving similar function or performance. Examples of detailing patterns can be the selected material layers of exterior walls from a specific side of the building, adding windows shading, or the type of joints between walls and slabs, which has a major impact on the transmission of thermal energy and sound (Châteauvieux-Hellwig et al. 2021). Detailing rationale includes the context information necessary to apply such patterns, such as the element's relative position to the storey's entrance and building's orientation (taking into account its sun path during the different seasons). Such domain knowledge is beneficial when detailing design options or designing new projects. However, currently, detailing decisions are embedded in building models, and detailing rationale is implicit in the designers' minds, hindering their proper management and reuse.

This paper introduces a parametric building graph (PBG) to capture detailing patterns, including the geometric, semantic, topological relationships and the rationale behind them. Additionally, it proposes a framework for automatically transfer detailings from one design to another, using graph transformation systems. The development of the PBG was based on reviewing the currently existing graph representations in the Architecture Engineering and Construction (AEC) industry.

The paper is organized as follows: Section 2 discusses the background and related work. Section 3 presents the research methodology, categorizes existing graph representations in the AEC industry, and proposes a practical framework for transferring detailing patterns. A feasibility study is presented and discussed in Section 4. Finally, Section 5 summarizes our progress hitherto and gives an outlook for future research.

2 Background and Related Work

2.1 Graph representations in the AEC industry

For more than a decade, graph structures were used in the AEC industry for various use-cases, including path planning (Rüppel et al. 2010; Hamieh et al. 2020), retrieval of similar designs (Langenhan et al. 2013), integration of heterogeneous building models (Hor et al. 2016), and encoding or engineering knowledge (Vilgertshofer & Borrmann 2017). Graphs structures are popular in the different domains due to their ability to represent complex relationships, which is the case in BIM (Isaac et al. 2013).

According to the graphs developed in the BIM domain, graphs include nodes representing building elements, in some cases their properties as well, and edges represent the relationships between them (Khalili & Chua 2015; Denis et al. 2017; Donato 2017; Ismail et al. 2018). Depending on the use-case, graphs could be as simple as raw nodes and edges, or attributed, where nodes and edges hold attributes (key-value pairs). The existing graph representations will be discussed in more detail in Section 3, where a categorization of these efforts is provided.

2.2 Computational design synthesis and graph rewriting

The field of Computational Design Synthesis (CDS) aims to formally describe design knowledge. Graphs structures are computationally well supported and capable of describing modular

product models. The concept of graph rewriting is described as a production system based on the combination of nodes and edges and their transformation rules (Helms et al. 2009).

Graph rewriting systems (GRS) are prevalent in capturing real-world and engineering design rules to synthesize design solutions (Chakrabarti et al. 2011). Rewriting systems are being investigated for more than a decade on formalizing design space of multiple domains, including mechatronic products (Helms et al. 2009), automotive powertrains (Helms & Shea 2012), multi-scale shield tunnel products (Vilgertshofer & Borrmann 2017), layout generation of architectural designs (Ruiz-Montiel et al. 2013) and evaluation of the connectivity of design solutions (Donato 2017). Performing graph rewriting requires three main parts: an original graph, a transformation subgraph, and a set of logical rules that match a particular subgraph pattern and perform a set of operations, including altering, deleting, or replacing nodes, edges, and their attributes. The result is an updated graph, where each matched pattern from the original graph is modified according to the logical rules.

The proposed framework in this paper for transferring detailing patterns from one building model to another makes use of GRS. The building model is represented as a graph, and the detailing decision is represented as a subgraph. Then, based on defined matching and rewriting patterns, the rewriting system produces a detailed graph of the building model.

3 Methodology

The hypothesis of this paper is divided into two main parts: (1) detailing patterns, including building information and the rationale behind them, can be captured using a graph representation. (2) GRS are capable of automatically transferring detailings between models through automatically generated rewriting rules.

3.1 Proposed approach for capturing and transferring detailing decisions

The proposed methodology is illustrated in Figure 2. First, designers formulate a detailing pattern, through a BIM-authoring tool, by selecting building elements, spaces, their relationships, and context information. When formulating a pattern, designers specify which information belongs to detailing and which belongs to reasoning when to apply it.

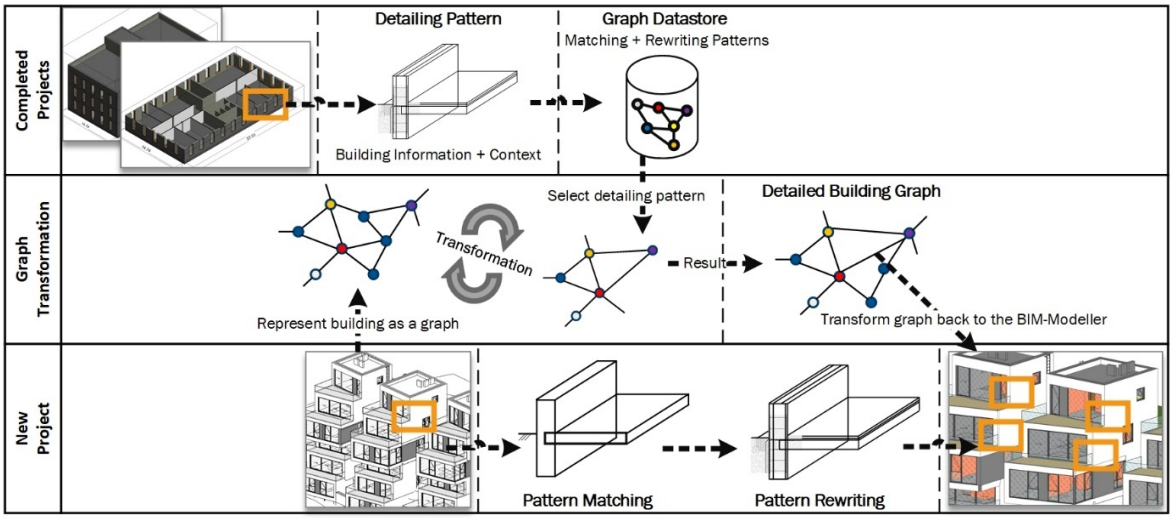

Figure 2. Proposed approach for capturing and transferring detailing decisions between models

A detailing pattern could include information of one or multiple elements. Such information includes: (1) geometric representation, such as shape, material layers, and junction types, (2) semantic information, represented with properties that include fire rating, load bearing, etc., (3) context information, describing relations to the nearby elements and the corresponding storey, building, or site. Examples of this context information could be the bounding room types, adjacent and accessible room types, distance from the entrance, or side of the building. The formulated

pattern is transformed into a graph representation and stored as a template in a data store. In the end, a detailing pattern can be described as:

$$Detailing\ Pattern = Matching\ Pattern + Rewriting\ pattern \quad (1)$$

Where the matching pattern finds the corresponding elements and the rewriting pattern specifies which nodes, edges, and attributes should be added, updated, or deleted. When designers detail a new design option or a new project, they can browse and select one of the stored patterns. As described before, transferring a detailing pattern to another model is based on GRS. Hence, the new model has to be transformed into a graph representation to apply the selected pattern on it. Applying the detailing pattern involves finding all its matches within the model graph. Then its corresponding nodes, edges, and their properties will be transformed with information from the rewriting pattern, producing a detained BIM model graph. Finally, the detail graph is transformed back into a BIM model inside the BIM-authoring tool.

3.2 BIM-authoring tool and detailing decisions

An essential step for storing and applying a detailing pattern is the transformation of building information from the BIM-authoring tool to a graph, and then back to the BIM-authoring tool after the application of detailing. In this paper, we selected Autodesk Revit[1] as a BIM-authoring tool since its API provides the ability to collect all the necessary information about building elements and their topological dependencies.

Currently, BIM-authoring tools provide various kinds of analysis[2] and advanced detailing information[2] in a parametric way. Accordingly, practitioners are provided with a user-friendly interface for detailing their models and specifying geometric constraints. Such capabilities were leveraged by researchers for multiple purposes, including performing automatic code compliance checking through the definition of calculated parameters (Patlakas et al. 2018). As designers use the functionalities offered by the BIM-authoring tool to develop their models, we have evaluated all the possible actions a designer can perform to detail a building model. This helps in confining the scope of this research, providing a practically applicable approach. Figure 3 shows a categorization of the possible detailing decisions. There are three main categories, geometry, semantics, and joints/connections. Geometrically, a designer can add a new geometric element (like placing a wall) as well as modify the representation via modelling or changing geometric parameters. Modifying the representation via modeling involves manipulating the element's type by refining or adding geometric shapes and parts.

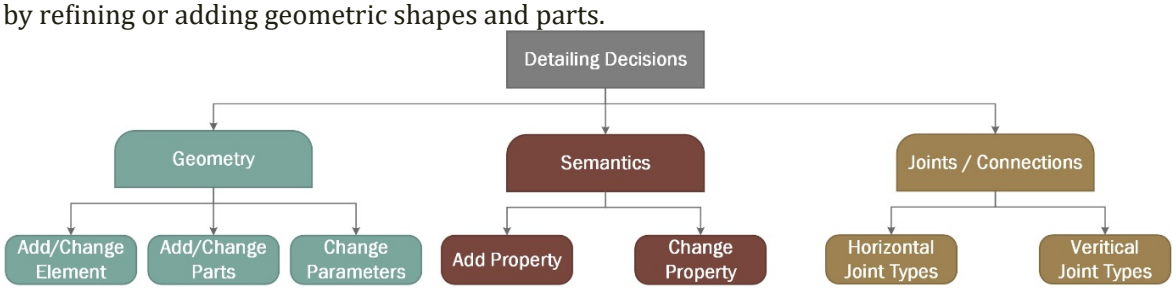

Figure 3. An overview of the available detailing decisions in BIM-authoring tools

Additionally, BIM-authoring tools provide functionalities assisting the modification of the geometry through manipulating a set of parameters using their user interface, such as adding a new material layer with a specific thickness to a wall. The category of semantics is straightforward; a designer can add, modify, or delete a property. Finally, there are two main approaches for joining building elements, either horizontally (e.g., when joining two walls) or vertically (e.g., when joining a wall and a slab), where each has a set of possible options (enumeration).

[1] https://www.autodesk.com/products/revit/overview
[2] https://autode.sk/3rWjXc3 | https://autode.sk/3dEYcs1 | https://autode.sk/2RiNksB

3.3 Categorization of graph representations

Based on our literature review, existing graph representations in the AEC industry can be categorized into four groups:

(1) **Space connectivity graphs**: spaces are represented as nodes, and edges represent either or both of the accessibility and adjacency between the different spaces. Connectivity graphs were used for evaluating similarity between designs, in a sense of a fingerprint (Langenhan et al. 2013; He et al. 2018), evaluating design quality (Donato 2017), reasoning about disability mobility (Strug & Ślusarczyk 2017), emergency path planning (Rüppel et al. 2010; Ismail et al. 2018), and security analysis (Porter et al. 2014).

(2) **Navigation graphs**: for the purpose of simulating pedestrian's behavior or navigating robots and drones, a space graph is not sufficient. Therefore, a finer graph representation is necessary, including additional special nodes representing visibility points (Kneidl et al. 2012) or navigation goals and interaction with the environment (Al Hattab & Hamzeh 2018; Dubey et al. 2020).

(3) **IFC model graphs**: multiple researchers have investigated transforming the IFC building models into graph representations (Khalili & Chua 2015; Ismail et al. 2018; Exner et al. 2019). The resultant nodes do not only represent building elements, but also their geometric representations, material layers, and more since the IFC schema is substantially expanding with every new release to support additional use-cases [3]. In a similar sense, ontology approaches were investigated in providing building representations for the purpose of seamlessly exchanging BIM models through web services, such as the Building Topology Ontology (BOT) (Rasmussen et al. 2019).

(4) **Knowledge representation graphs**: multiple researchers have leveraged graphs for formalizing knowledge (Solihin & Eastman 2016; Vilgertshofer & Borrmann 2017) and linking heterogeneous data models (Hor et al. 2018), where a customized graph representation or the combination of multiple graph structures is used. The same applies to parametric models, where a specific logic is embedded within the different graph nodes.

The category closest to our needs for capturing and transferring detailing decisions is the IFC model graphs. However, the IFC schema is a strict representation intended to be implemented by BIM software vendors to provide a neutral medium for exchanging BIM models. Accordingly, IFC is based on a relational model representation, where it includes objectified relationships and properties. Such representation is not flexible enough for capturing the custom detailing patterns and is not optimal for the usage as a graph "as-is"; various simplifications and manipulations are required. Additionally, transferring detailing patterns back to the BIM-authoring tools is an essential requirement for this research. Therefore, a simple graph structure that is capable of representing spaces and building elements, including their detailing, is necessary.

3.4 Graph representation for capturing detailing patterns

Based on investigated detailing decisions and reviewed graph representations in the AEC industry, the need for a new and simplified graph representation that is capable of capturing detailing patterns was identified. The meta-model of the proposed graph representation is shown in Figure 4. A graph comprises at least one node and can include multiple edges. The class *ElementNode* is the parent node class that holds attributes describing a node's identity as well as its corresponding matching and rewriting patterns. In terms of geometric representation, the geometric parameters (including the geometric parts, their properties, and order), as well as the bounding box of each element, are captured.

As inheritance nodes from the *ElementNode*, the *GeometricElementNode* and *SpatialContainerNode* capture an additional set of properties. Here we differentiate between geometric elements that are simple (e.g., a column or a one-layered wall) and assembly (e.g., multi-layered wall or multi-part components). The *SpatialContainerNode* represents any space that has an implicit representation, like a storey or a building, while the *SpatialElementNode*

[3] https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/

represents the actual spaces, (e.g., modelled rooms and zones). There are three main types of edges for describing the relationships between the different kinds of nodes: (1) *ContainedIn* (directed edge), describes the relationship between geometric and spatial elements, where the direction specifies the containment's host, for example, a wall is *ContainedIn* a room, and an opening is *ContainedIn* a wall, (2) *IsAdjacent* (undirected edge), links adjacent spaces with each other, identifying their accessibility, and (3) *IsConnected* (undirected edge), describes the connections and joints between the geometric elements. The connection point between two elements is represented through the angle of their bounding boxes and a detailed connection position between their faces, using horizontal and vertical anchors and paddings.

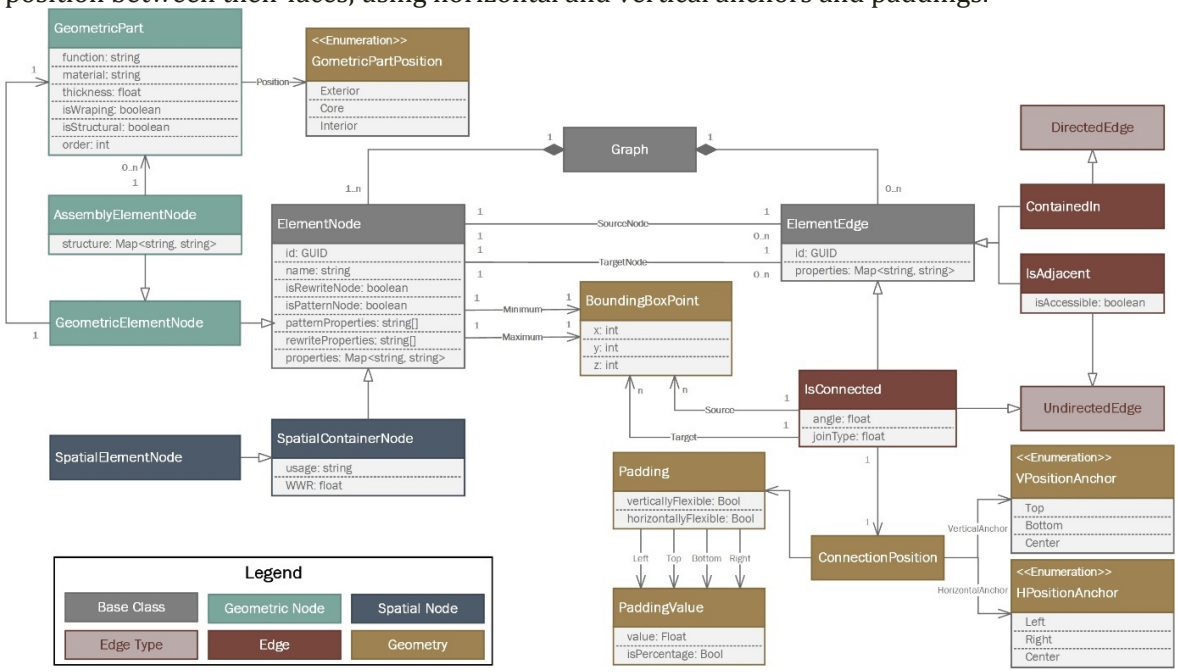

Figure 4. Parametric Building Graph (PBG): meta-model (UML diagram)

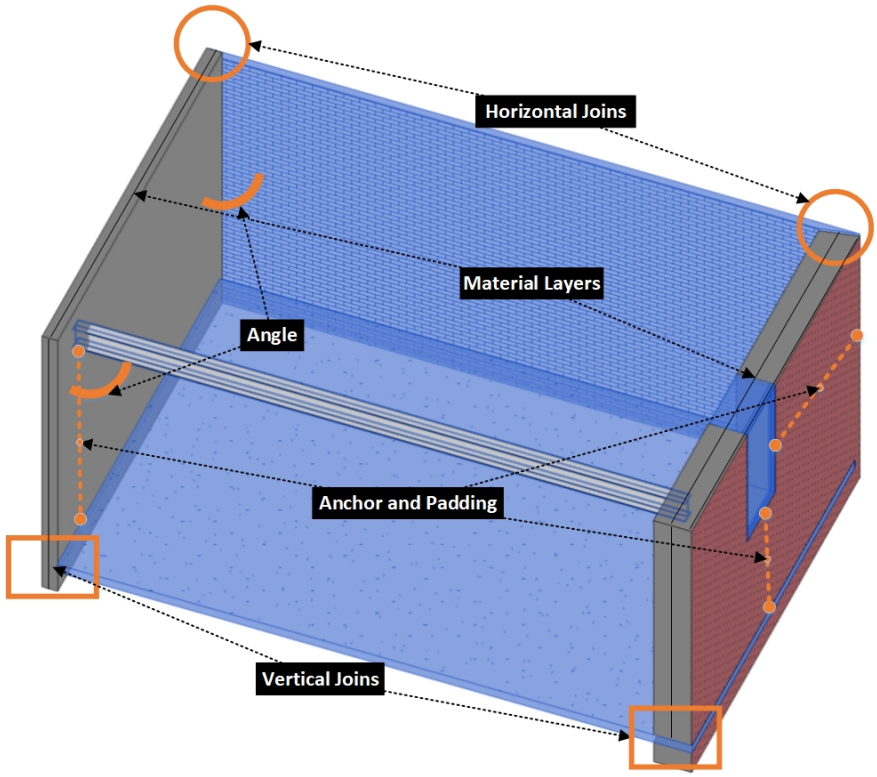

Figure 5. Illustration of the captured positions, connections, and joints between building elements

Figure 5 highlights multiple concepts that were discussed so far for describing the captured geometry. The horizontal and vertical joints show two different options for each. Additionally, when describing the connections between the bounding boxes of the elements, the angle, anchor, and padding are measured to describe their relative position as either a raw value or percentage.

3.5 Graph rewriting systems for transferring detailing patterns

As described previously, we propose transferring detailing patterns using GRS. Graph transformations are based on declarative rules that specify a set of modifications of graphical structures. The essential process of performing graph transformations matches a pattern graph within a large graph (a.k.a., host graph) and then applying graph modifications. Subgraph matching is known as an NP-complete problem (Geiß et al. 2006). A popular algorithm for overcoming such a problem is *Search Plan* (Batz et al. 2007). *Search Plan* is a heuristic optimization algorithm where a sequence of primitive matching operations is performed. In this regard, a cost value is assigned to the different operations. Accordingly, such algorithms perform matching gradually during runtime on the corresponding host graph.

GRS make use of heuristic pattern matching algorithms to perform their graph transformations. The configuration of such transformation is represented through a set of *Rewriting Rules.* A rule consists of four main parts (see Figure 5). Two parts match a pattern according to a set of nodes, edges, as well as logical checks (including if-else conditions) that are performed on their properties. For example, a pattern of a wall separating two rooms, where the wall's material is wood and the area of one of the rooms is larger than or equal to 20sqm. To prevent encountering infinite updates and loops through the matched patterns, a *Negative Pattern* is defined through a graphlet that acts as a skipping criterion. Typically, a negative pattern includes a graphlet of the matching pattern after rewriting. When a pattern passes the negative pattern check, a *Rewriting Pattern* is applied on it. A rewriting pattern includes a description of what modifications will be performed, including adding, deleting, and modifying nodes, edges, and their properties.

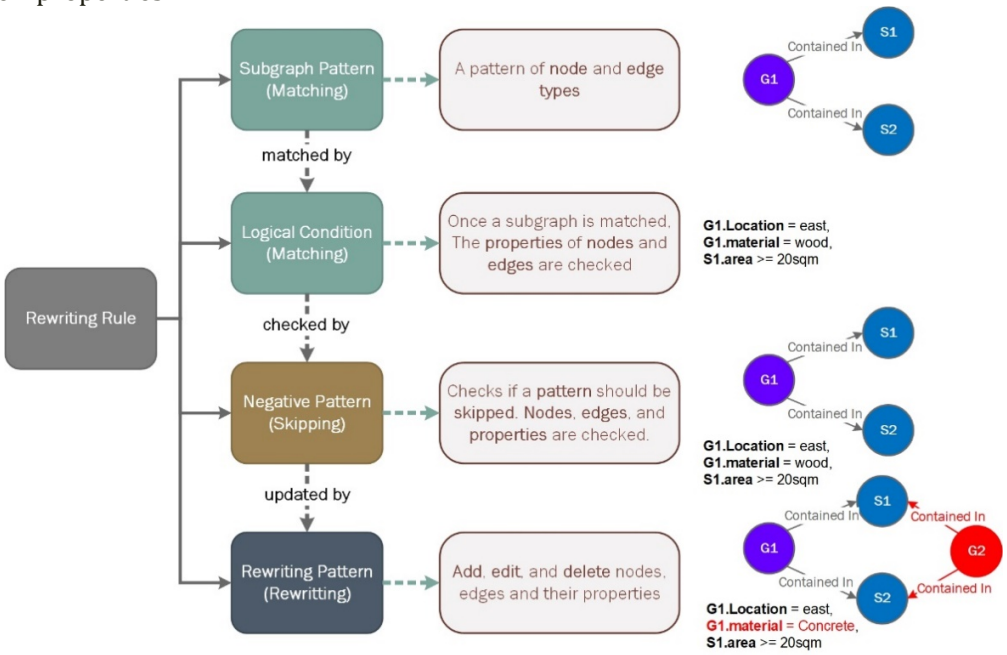

Figure 6. Graph rewriting system: structure of a rewriting rule

Our approach proposes an automatic generation of rewriting patterns from the captured detailing pattern, which is specified through the user interface of the BIM-authoring tool. Accordingly, the main advantage of this automation step is to reduce the burden for engineers and architects of having additional knowledge in formally defining rewriting rules as a prerequisite of using the GRS.

4 Feasibility Study

To evaluate the applicability of the proposed approach, the API of Autodesk Revit was used to export the building information into the developed graph representation. Additionally, we have selected GrGen.NET (Jakumeit et al. 2010) as a GRS since it provides an API for interacting with its algorithmic kernel and its libraries can be integrated within a Revit plugin. GrGen uses the *Search Plan* algorithm to perform subgraph matching. Accordingly, matching a detailing pattern is achieved through a sequence of search operations for the individual matching nodes and edges, taking into account the structure of the building graph during runtime.

With the help of a plug-in developed for Revit, a selected element, like a wall, is visualized alongside its corresponding properties, material layers, joints, and connections to adjacent elements as well as rooms. Using the user interface, it is possible to select which properties and nodes belong to the matching pattern or the rewriting pattern. Then, the formulated detailing pattern can be stored in order to be transferred later to another building model. Figure 7 shows a snapshot of the developed plugin inside Revit. Here, the detailing pattern is formalized for the selected exterior wall, where it is bounding a room with a *bedroom* as usage, this relationship is selected as part of the matching pattern, and two windows, contained in the wall, are selected as part of the rewriting pattern. The relative position anchor and padding can be specified through the position button beside each row. The formulation of the matching and rewriting patterns combines more information about the selected elements under the other tabs.

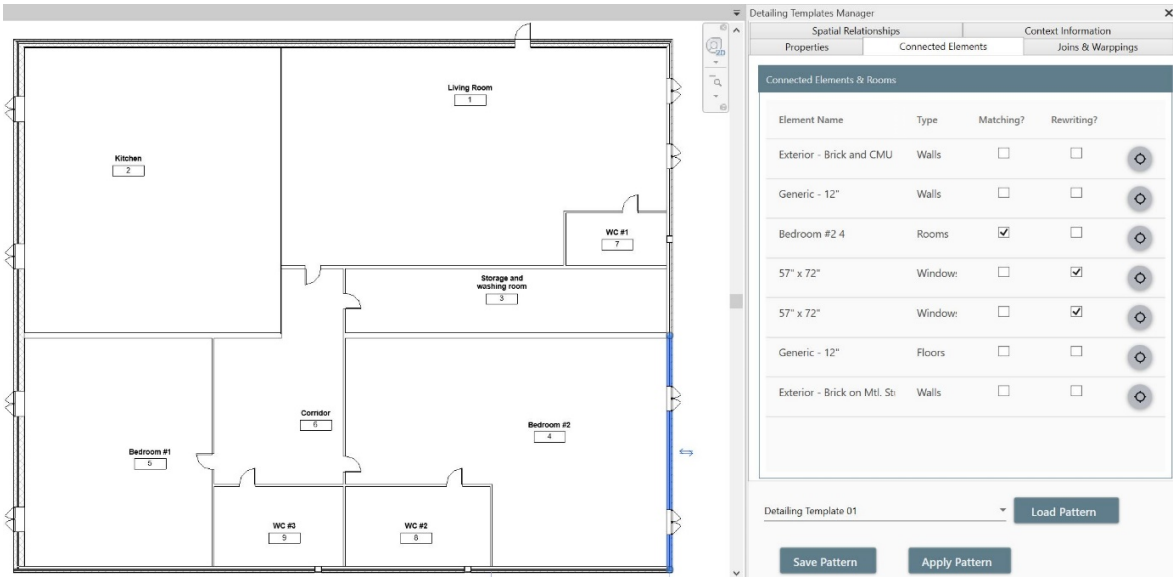

Figure 7. Prototype: Autodesk Revit plugin for capturing and transferring detailing patterns

In this study, we evaluated the automatic transfer of the detailing decisions discussed in Section 3.2 on multiple building designs. As a result, we were able to successfully generate a detailed building graph for all the categorized detailing decisions, except automatically editing the geometric shapes of a building element type (a.k.a., family). Automatically detailing a type's shape requires extending the proposed graph structure for capturing the geometric operations in more detail.

Figure 8 presents an example of the graph representation and an automatically generated rewriting rule. The building graph represents a storey with three rooms that are surrounded by simple and assembly walls. The detailing pattern tries to find a pattern of two not accessible spatial elements, where the area of one of the rooms is bigger than 20sqm, then places a door in the wall separating them, according to their relative position, and changes the wall's material to *Brick*. Such detailing pattern searches for a matching subgraph of nodes and edges (same structure) and then checks whether the area is larger than 20sqm. Before transferring the detailing pattern, the *kitchen* and *living room* were separated by a *Concrete* wall, and there was

no accessibility between them. Once the pattern was matched and rewritten, a door was added to the separating wall, and the wall's material was changed to *Brick*.

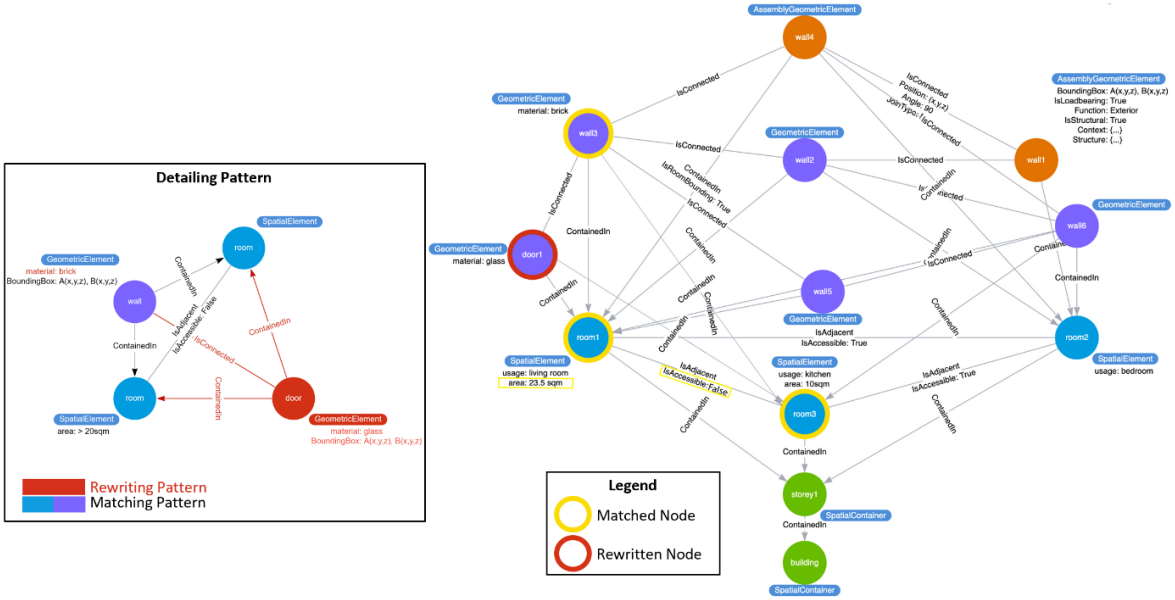

Figure 8. An example of a graph representation and an automatically generated rewriting rule

5 Conclusions and Future research

Design and detailing decisions highly influence the resultant building's performance and compliance with regulations. Architects and engineers reuse their successful experiences to transfer those decisions to new projects and design variants. This paper introduced a simplified and parametric building graph that is capable of capturing detailing patterns, including building information and the rationale behind them. Additionally, a framework that is based on graph transformation systems was proposed for automatically transferring detailing decisions from one design to another. Through evaluation of the implemented prototype, the proposed approach was able to handle multiple detailing decisions, including adding elements, modifying elements' geometry through parameters, as well as manipulating their semantics.

As future work, modifying the element's type geometry will be investigated in detail. In this regard, the fundamental geometric operations will be captured and reproduced through graph theory techniques. Additionally, an extensive evaluation of the developed framework on different sizes and types of building projects will be investigated.

Acknowledgements

We gratefully acknowledge the support of the German Research Foundation (DFG) for funding the project under grant FOR 2363.

References

Abualdenien, J. & Borrmann, A. (2019) A meta-model approach for formal specification and consistent management of multi-LOD building models. *Advanced Engineering Informatics* **40**, 135–153.

Al Hattab, M. & Hamzeh, F. (2018) Simulating the dynamics of social agents and information flows in BIM-based design. *Automation in Construction* **92**, 1–22.

Batz, G. V., Kroll, M. & Geiß, R. (2007) A first experimental evaluation of search plan driven graph pattern matching. In: *International Symposium on Applications of Graph Transformations with Industrial Relevance*, pp. 471–486.

Chakrabarti, A., Shea, K. & Stone, R. et al. (2011) Computer-based design synthesis research: an overview. *Journal of Computing and Information Science in Engineering* **11** (2).

Denis, F., Temmerman, N. de & Rammer, Y. (2017) The potential of graph theories to assess buildings' disassembly and components' reuse: How building information modelling (BIM) and social network

analysis (SNA) metrics might help Design for Disassembly (DfD). In: *Proceedings of the HISER International Conference*.
Donato, V. (2017) Towards design process validation integrating graph theory into BIM. *Architectural Engineering and Design Management* **13** (1), 22–38.
Dubey, R. K., Khoo, W. P., Morad, M. G., Hölscher, C. & Kapadia, M. (2020) AUTOSIGN: A multi-criteria optimization approach to computer aided design of signage layouts in complex buildings. *Computers & Graphics* **88**, 13–23.
Exner, H., Abualdenien, J., König, M. & Borrmann, A. (2019) Managing Building Design Variants at Multiple Development Levels. In: *Proc. of the 36th International Council for Research and Innovation in Building and Construction (CIB W78)*, Newcastle, UK.
Geiß, R., Batz, G. V., Grund, D., Hack, S. & Szalkowski, A. (2006) GrGen: A fast SPO-based graph rewriting tool. In: *International Conference on Graph Transformation*, pp. 383–397.
Hamieh, A., Makhlouf, A. B., Louhichi, B. & Deneux, D. (2020) A BIM-based method to plan indoor paths. *Automation in Construction* **113**, 103120.
He, T., Zhang, J., Lin, J. & Li, Y. (2018) Multiaspect similarity evaluation of BIM-based standard dwelling units for residential design. *Journal of Computing in Civil Engineering* **32** (5), 4018032.
Helms, B. & Shea, K. (2012) Computational synthesis of product architectures based on object-oriented graph grammars. *Journal of Mechanical Design* **134** (2).
Helms, B., Shea, K. & Hoisl, F. (2009) A framework for computational design synthesis based on graph-grammars and function-behavior-structure. In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 841–851.
Hor, A. E., Gunho, S., Claudio, P., Jadidi, M. & Afnan, A. (2018) A SEMANTIC GRAPH DATABASE FOR BIM-GIS INTEGRATED INFORMATION MODEL FOR AN INTELLIGENT URBAN MOBILITY WEB APPLICATION. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* **4** (4).
Hor, A. H., Jadidi, A. & Sohn, G. (2016) BIM-GIS integrated geospatial information model using semantic web and RDF graphs. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci* **3** (4), 73–79.
Isaac, S., Sadeghpour, F. & Navon, R. (2013) Analyzing building information using graph theory. In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, p. 1.
Ismail, A., Strug, B. & Ślusarczyk, G. (2018) Building knowledge extraction from BIM/IFC data for analysis in graph databases. In: *International Conference on Artificial Intelligence and Soft Computing*, pp. 652–664.
Jakumeit, E., Buchwald, S. & Kroll, M. (2010) Grgen. net. *International Journal on Software Tools for Technology Transfer* **12** (3), 263–271.
Khalili, A. & Chua, D. H. K. (2015) IFC-based graph data model for topological queries on building elements. *Journal of Computing in Civil Engineering* **29** (3), 4014046.
Kneidl, A., Borrmann, A. & Hartmann, D. (2012) Generation and use of sparse navigation graphs for microscopic pedestrian simulation models. *Advanced Engineering Informatics* **26** (4), 669–680.
Langenhan, C., Weber, M., Liwicki, M., Petzold, F. & Dengel, A. (2013) Graph-based retrieval of building information models for supporting the early design stages. *Advanced Engineering Informatics* **27** (4), 413–426.
Patlakas, P., Livingstone, A., Hairstans, R. & Neighbour, G. (2018) Automatic code compliance with multi-dimensional data fitting in a BIM context. *Advanced Engineering Informatics* **38**, 216–231.
Porter, S., Tan, T., Tan, T. & West, G. (2014) Breaking into BIM: Performing static and dynamic security analysis with the aid of BIM. *Automation in Construction* **40**, 84–95.
Rasmussen, M. H., Lefrançois, M., Schneider, G. F. & Pauwels, P. (2019) BOT: the building topology ontology of the W3C linked building data group. *Semantic Web* (Preprint), 1–19.
Ruiz-Montiel, M., Boned, J., Gavilanes, J., Jiménez, E., Mandow, L. & PéRez-De-La-Cruz, J.-L. (2013) Design with shape grammars and reinforcement learning. *Advanced Engineering Informatics* **27** (2), 230–245.
Rüppel, U., Abolghasemzadeh, P. & Stübbe, K. (2010) BIM-based immersive indoor graph networks for emergency situations in buildings. In: *Proceedings of the International Conference on Computing in Civil and Building Engineering*.
Schneider-Marin, P. & Abualdenien, J. (2019) A framework to facilitate an interdisciplinary design process using BIM. In: *Proc. of the 31th Forum Bauinformatik*, Berlin, Germany.
Solihin, W. & Eastman, C. M. (2016) A knowledge representation approach in BIM rule requirement analysis using the conceptual graph. *ITcon* **21**, 370–401.
Strug, B. & Ślusarczyk, G. (2017) Reasoning about accessibility for disabled using building graph models based on BIM/IFC. *Visualization in Engineering* **5** (1), 1–12.
Vilgertshofer, S. & Borrmann, A. (2017) Using graph rewriting methods for the semi-automatic generation of parametric infrastructure models. *Advanced Engineering Informatics* **33**, 502–515.