Presentation of an optimized process for generating analysis models in structural engineering using Graph Neural Networks

Toni Nabrotzky[1]

[1]IHBB | Institut für Hochbau, Baukonstruktion und Bauphysik, Leipzig University of Applied Sciences, Karl-Liebknecht-Straße 143, 04277 Leipzig, Germany

E-mail(s): toni.nabrotzky@htwk-leipzig.de

Abstract: Building Information Modeling (BIM) methodology is increasingly used in the construction industry. When this method is utilized for the structural integrity analysis of architectural models, these models must be converted into finite element models. However, automated processes of model conversion are prone to errors, as relevant connections between BIM objects are often missing. Hence, structural engineers often prefer the manual creation of the analysis model in practice. The aim of this paper is to introduce new methods for an efficient and collaborative workflow for automated model conversion. To this end, the use of an AI system in the form of a Graph Neural Network (GNN) is proposed. This system aims to recognize the load-bearing components in a BIM model through semantic enrichment and to establish their relationships based on load transfer in the form of a graph model. Incorrect connections or missing links between building elements are identified using a GNN approach to repair these issues and create correct and computationally efficient structural analysis models. Through these optimized processes, the need for individuall model adaption is expected to be reduced in the future.

Keywords: Building Information Modeling (BIM), Industry Foundation Classes (IFC), structural engineering, machine learning

Erschienen in Tagungsband 35. Forum Bauinformatik 2024, Hamburg, Deutschland, DOI: 10.15480/882.13522

1 Introduction

The ongoing development of Building Information Modeling (BIM) in the construction industry is reshaping work processes and tools employed therein. While conventional practices often rely on 2D plans for the exchange of project-specific information, BIM offers the capability to utilize enriched 3D models for collaboration within a project [1]. This transition to digital planning and construction introduces a range of novel benefits, as it integrates both geometric and alphanumeric data into the model. Consequently, BIM models can be directly leveraged for various analysis and computational tools [2]–[4].

One application area is structural analysis. Currently data exchange is limited due to the different geometric representations of the architectural model and the analysis model. Conversion is achieved through hardcoded rules, which require significant manual effort to create. The resulting rule sets are often tailored to specific structures and are difficult to apply to other projects without risking errors in translation. Sibenik et al. [5] developed the program *redDim*, which interprets the geometry of a structure and transforms it into an analysis model by reducing its dimensionality. Additionally, Jia et al. [6] proposed a method for automatically generating Finite Element Models from BIM models. This method extracts necessary information from the BIM model and constructs the corresponding element types using an automatic decision-making mechanism.

However, the methods published so far do not allow for fully automatic transformations of the models and still require the manual creation of construction rules for interpreting the ontology. Additionally, some methods are designed for calculations with volumetric models (three-dimensional elements such as tetrahedrons and hexahedrons) and are not, or only partially, suited for the analysis of 1D and 2D elements. In the construction industry, however, efficient methods that are as fully automated as possible and utilize simplified calculation procedures with abstracted components are economically advantageous.

Artificial Intelligence (AI) has significant potential for improving on these methods. By training on large datasets, AI algorithms can identify patterns and trends that humans may overlook or require more time to recognize [7]. Therefore, advanced techniques such as Deep Learning (DL) and Machine Learning (ML) are employed for various purposes, including predicting analysis calculations [8], [9] and automating routine tasks [10]. For instance, Bloch et al. [11] investigated the use of graph-based learning algorithms instead of hardcoded rules for automated structural inspections. Their research demonstrates that graph-based models generally outperform classical ML models in classifying structural elements. They recommend incorporating not only geometric but also topological information into the models [12].

In this paper we develop a new AI-powered workflow to further advance data exchange between structural design and architectural design. This AI-assisted approach aims to simplify the model transformation from the architectural model to the analysis model for structural design. Particularly in early planning phases, this methodology supports data exchange, as a large number of models can be automatically converted and analyzed, for example, in variant studies. It should be noted that this paper aims to present the conceptual design. A comprehensive analysis of the accuracy of the AI system and the parameters to be selected for AI training is still pending and will be addressed in future work. This new workflow can be divided into three phases (see Figure 1):

Extract Information: The basic concept of the new workflow involves extracting all load-bearing structural elements from an IFC model along with their metadata. In this context, metadata includes all non-structural information.

Graph Construction: Subsequently, this feature list is used to create a spatial property graph, with each structural element represented as a node. The extracted metadata is attached to these nodes as

node features. Using the concept of link prediction, a Graph Neural Network (GNN) is employed to predict which elements can transfer loads to each other and represent this relationship as an edge.

Repair of analysis model: With the understanding of which elements interact with each other during load transfer, this knowledge base is utilized to repair the associated analysis model if not all beam and surface elements are seamlessly connected. For this workflow, it is assumed that the analysis model is not in a proprietary format but in matrix format or in tabular form, such as the SAF format [13]. Therefore, a matching algorithm assigns the structural elements to the nodes in the graph. By linking the entities, it can then verify whether the nodes and edges of those elements are related to each other are on a common level. If this is not the case, it adjusts the positions of the elements accordingly.

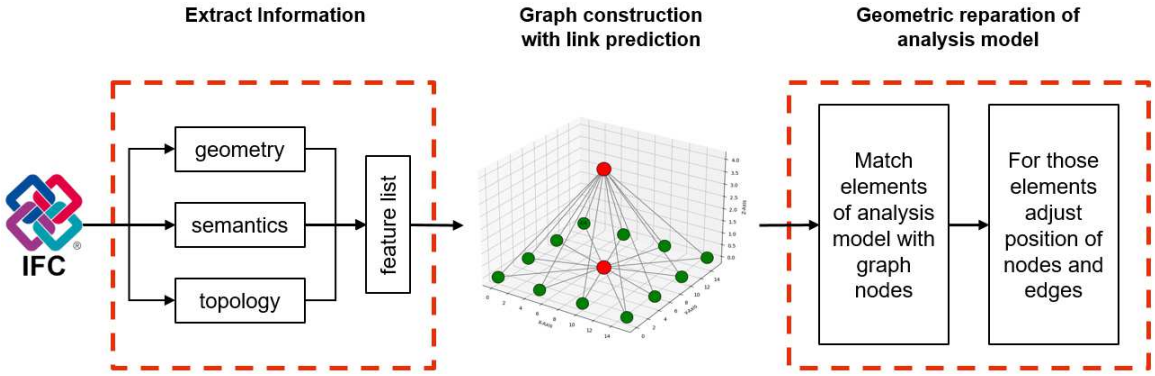

Figure 1: Presentation of the optimized workflow of the proposed framework. First, extract necessary information. Second, employ a Graph Neural Network (GNN) to identify relationships between the building elements. Third, align the nodes with your analysis model to correct the position and alignment of the model.

2 Framework for optimizing analysis models

2.1 Creating a database

Since a large number of labeled models is not available, we propose the use of parametrically generated models. Different value ranges can be defined for individual parameters, which are then combined into a parameter set. This allows for the automated creation of a large database comprising several hundred models. By further dividing the models into submodels, the existing quantity of training data can be maximized. However, this approach is only useful within certain limits, as the parametric construction of the building results in many similar models within the database. This poses the risk of overfitting, where the AI system becomes too closely adapted to the training data and fails to learn the systematics behind its actual task. To counteract this, it is recommended to augment the training data with synthetically generated data, as successfully implemented by Bloch et al. in their experiments [11].

For the upcoming training of the AI system, a parametric training model, as shown in Figure 2 will be utilized. This model consists of a simple structure composed of slabs and columns, which was programmed using Rhino and the visual programming environment Grasshopper with the Visual-ARQ plugin [14]. The following parameters will be varied:

- Dimensions in the x and y directions
- Floor height
- Number of floors
- Number of columns per floor
- Position of columns per floor

After conducting the initial training runs, the dataset used must be checked for overfitting. If overfitting is detected, the proportion of the parametric model in the training data needs to be reduced. Simultaneously, the dataset must be expanded to include additional models with different geometries, materials, and usage types. To achieve this goal, the parametric example model can be extended to include additional component types (walls, beams, etc.) and additional floors. Other models that should be included as training data can be real projects from industry partners who provide some IFC models for training the AI system.

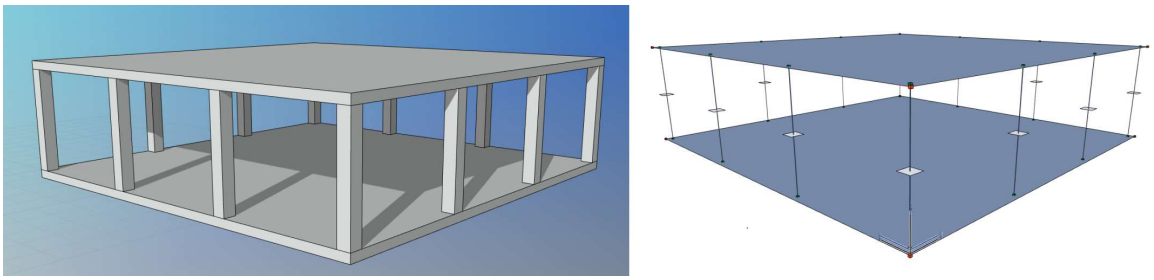

Figure 2: Parametric example model for generating training data. The structural model, displaying all load-bearing elements, is shown on the left. The analysis model is presented on the right.

2.2 Selection of relevant information

Before beginning the extraction of metadata from the IFC model, the relevant information for this purpose must be determined. In addition to alphanumeric attributes, the basis for training an ML model includes geometric data, material data, loading data, physical data, or the intended use and functionality. Since this workflow is primarily intended for application in early planning phases (1-3), it cannot be assumed that data on loads, stresses, and internal forces are already present in the model. Further data, such as building physics parameters or fire protection information, play a subordinate role in determining and analyzing the load-bearing elements. Therefore, for the extraction of metadata, the focus initially lies on an information set consisting of three essential aspects:

- Alphanumeric attributes
- Geometric information
- Material data

Alphanumeric attributes are used to describe each building element. In the base class 'IfcRoot', the attributes 'GUID' (IfcGloballyUniqueId) and 'Name' are created and inherited by all subclasses. All components with a physical or functional representation are described by the class 'IfcProduct'. At a deeper implementation level, these components are then assigned an abstract 'IfcType,' such as 'IfcWallStandardCase' or 'IfcSlab'. This 'IfcType' is also extracted.

Geometric information is essential for forming the structural system. It describes, for example, the position (the origin coordinate) of the element or its extent in the x, y and z directions. Orientation and rotation are indicated by direction vectors, showing the direction in which a component is aligned. References to both global and local coordinate systems can be established. Additionally, symmetry or shape can be described under this category, indicating whether the cross-section is square, circular, or otherwise defined. Spatial distances to other components, which are important for collision checks, for example, can also be derived from the model.
Material data serves to identify the type of material present. Examples of these materials include concrete, steel, wood, and masonry. Additional physical properties are not included in the training. However, in future training sessions of the ML system, it should be investigated what influence these properties have on link prediction. It may be conceivable to expand the database to include material density, thermal conductivity, fire resistance, or ecological data.

2.3 Extraxtion of data

This section describes the algorithm for this process step. We will use the Python program library *ifcopenshell* [15] for parsing the IFC data. With this library, all entities with the class IfcProduct and an existing physical representation will first be filtered. Next, the property sets of these elements will be read and checked to see if the respective component has the property 'LoadBearing' = 'True'. If so, the attributes described above will be extracted.

Since a physical building element in an IFC model can be described by different representations, such as 'Brep', 'MappedRepresentation' or 'SweptSolid', we will initially make the simplified assumption that all components are horizontally or vertically oriented. This allows us to describe the entities using their bounding box. The exact determination of all vertices can be implemented at a later stage when the training data includes more complex models. For material data, we will initially only query which material type was defined for the element. Subsequently, the collected information will be saved as a JSON file.

2.4 Graph Neural Network

Graph Structure: The proposed graph shall be designed in form of a spatial property graph. This type of graph integrates spatial data with traditional graph properties, enabling the representation and analysis of both attribute-based and spatial relationships. Formally it can be described as: $G = (N, E, C, M)$ where:

- N is the set of nodes (vertices) in the graph
- E is the set of edges (links) connecting the nodes
- C is the set of coordinates, representing the origin coordinates (spatial position) of the nodes
- M is the set of metadata (properties) associated with the nodes and edges.

The aim of this workflow is for a GNN to automatically generate and represent spatial property graphs precisely. To achieve the goal of accurate predictions, consideration must be given to how the structure of the graph should look and how the GNN should ideally be trained. The proposed graph comprises all load-bearing elements as nodes, their interaction in load transfer as edges, their spatial origin as

coordinates and their metadata as node features. Using the program library *networkx* [16], an empty undirected graph will be initialized. Information about nodes and their respective node features can then be extracted from the JSON file. It is important to note that the data from the JSON file must first be converted into numerical feature vectors for a GNN to process the extracted information. Using the methodology of Link Prediction, the GNN should then predict the relationships between the nodes. Subsequently, the predicted connections between the nodes by the GNN can be implemented as well.

In Figure 3 the process of data extraction from the IFC model and the subsequent initialization of the graph is depicted. It illustrates the detailed processes of the first two steps 'Extraxt information' and 'Graph construction with link prediction' from Figure 1. First it shows how each element in an IFC file is examined. If the requirements that it is a physically present component defined as 'load-bearing' are met, the relevant information is extracted and stored in a JSON file. These data are then extracted, and the graph with its nodes and node features is created. The GNN completes this graph by predicting the connections. This graph serves as the working basis for the next step in the process, the geometric error correction of the analysis model.

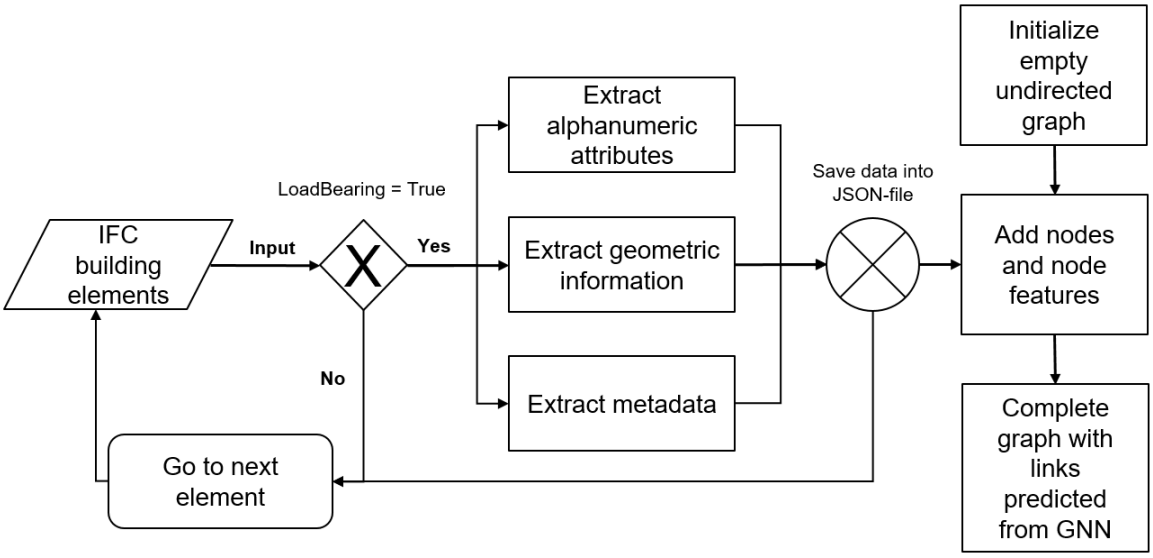

Figure 3: Process of creating spatial graph

Training the GNN: The planned GNN will be coded using the program libraries *PyTorch* and *PyTorch Geometric* [17], [18]. It will use the empty graph filled in with the information from the JSON files as input data, which need to be converted into an appropriate structure. The generated training data must be divided into training, validation, and test sets. Typical models for training GNNs include Graph Convolutional Networks (GCNs) or Graph Attention Networks (GATs). At this point, no methodology has been established on how the GNN should be trained. In later experimental phases, both architectures will be applied and tested to decide on one of the two variants.

In some IFC models, relationships are already stored using 'IfcRelationship'. It is important to note that some relationships may be redundant from the perspective of structural engineering, which could potentially confuse the GNN during the learning process. Therefore, it is necessary to investigate

whether training for efficient link prediction could benefit from the use of these data if they are available. In the application of this framework later on, it is conceivable that the existing connections are already integrated into the graph, and the GNN merely examines their plausibility and removes unnecessary connections besides its actual task of the classical link prediction.

2.5 Repair of the analysis model

The graph completed by the GNN forms the basis for the next steps, aimed at repairing an existing analysis model. A matching algorithm then determines the similarities between the elements from the analysis model and the nodes from the graph. The knowledge of the relationship between multiple elements is then used to repair geometric errors in the analysis model. This involves examining whether the nodes and edges of elements from the analysis model that are connected in the graph are on a common plane. If not, the coordinates of the affected points need to be adjusted. In the case of the example model, no errors occur due to the rule-based derivation, given its low complexity. However, when evaluating the effectiveness of the workflow in future work, the matrices of the analysis models will need to be manually manipulated to introduce errors at this stage. In further experiments with more complex models, this step will no longer be necessary, as the analysis models will not be directly computable as finite element models.

3 Outlook

The workflow presented in this paper is particularly intended for early planning phases, where structural engineering often does not yet have its own specialized model but needs to explore many design variants using the 3D model. In future experimental runs, it will be examined whether the selected relevant information is suitable for training the AI system and whether additional information should be included. However, only information that is typically available in the early planning stages should be integrated. Additionally, it needs to be validated how existing data from 'IfcRelationship' influences the training of the GNN. For the adjustment of elements, new sets of rules must be created. It is conceivable that components could be assigned specific priorities. When multiple elements need to be aligned, an algorithm could designate components with the highest priority as the fixed point to which all elements with lower priorities are adjusted. The workflow can then be examined for its functionality and effectiveness. It should be considered effective if the revised analysis models exhibit fewer errors than before and result in a computable finite element model.

Literature

[1] A. Borrmann, M. König, C. Koch, and J. Beetz, *Building Information Modeling: Technologische Grundlagen und industrielle Praxis*. Springer-Verlag, 2015.

[2] S. Azhar, "Building information modeling (BIM): Trends, benefits, risks, and challenges for the aec industry", *Leadership and Management in Engineering*, vol. 11, no. 3, pp. 241–252, 2011. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29LM.1943-5630.0000127.

[3] P. Bynum, R. R. A. Issa, and S. Olbina, “Building information modeling in support of sustainable design and construction”, *Journal of Construction Engineering and Management*, vol. 139, no. 1, pp. 24–34, Jan. 2013. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29CO.1943-7862.0000560.

[4] E. Krygiel and B. Nies, *Green BIM: Successful Sustainable Design with Building Information Modeling*, 1st ed. Wiley, 2008.

[5] G. Sibenik, I. Kovacic, V. Petrinas, and W. Sprenger, “Implementation of open data exchange between architectural design and structural analysis models”, *Buildings*, vol. 11, no. 605, 2021. [Online]. Available: https://www.mdpi.com/2075-5309/11/12/605.

[6] J. Jia, J. Gao, W. Wang, L. Ma, J. Li, and Z. Zhang, “An automatic generation method of finite element model based on bim and ontology”, *Buildings*, vol. 12, no. 11, 2022. [Online]. Available: https://www.mdpi.com/2075-5309/12/11/1949.

[7] S. Haghsheno, G. Satzger, S. Lauble, and M. Vössing, *Künstliche Intelligenz im Bauwesen*, 1st ed. Springer Vieweg Wiesbaden, 2024.

[8] R. E. Meethal, A. Kodakkal, M. Khalil, *et al.*, “Finite element method-enhanced neural network for forward and inverse problems”, *Advanced Modeling and Simulation in Engineering Sciences*, vol. 10, no. 1, p. 6, 2023. [Online]. Available: https://doi.org/10.1186/s40323-023-00243-1.

[9] D. Roznowicz, G. Stabile, N. Demo, D. Fransos, and G. Rozza, “Large-scale graph-machine-learning surrogate models for 3d-flowfield prediction in external aerodynamics”, *Advanced Modeling and Simulation in Engineering Sciences*, vol. 11, no. 1, p. 6, 2024. [Online]. Available: https://doi.org/10.1186/s40323-024-00259-1.

[10] A. Buruzs, M. Šipetić, B. Blank-Landeshammer, and G. Zucker, “Ifc bim model enrichment with space function information using graph neural networks”, *Energies*, vol. 15, no. 8, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/8/2937.

[11] T. Bloch, A. Borrmann, and P. Pauwels, “Graph-based learning for automated code checking - exploring the application of graph neural networks for design review”, *Advanced Engineering Informatics*, vol. 58, p. 102 137, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034623002653.

[12] G. Austern, T. Bloch, and Y. Abulafia, “Incorporating context into bim-derived data - leveraging graph neural networks for building element classification”, *Buildings*, vol. 14, 2024. [Online]. Available: https://www.mdpi.com/2075-5309/14/2/527.

[13] SAF-Documentation. (2024), [Online]. Available: https://www.saf.guide/en/stable/.

[14] VisualARQ. (2024), [Online]. Available: https://www.visualarq.com/de/.

[15] IfcOpenShell-Community. (2024), [Online]. Available: https://ifcopenshell.org/.

[16] NetworkX. (2024), [Online]. Available: https://networkx.org/.

[17] PyTorch. (2024), [Online]. Available: https://pytorch.org/.

[18] PyTorchGeometric. (2024), [Online]. Available: https://www.pyg.org/.