Pre-trained language model based method for building information model to building energy model transformation at metamodel level

Zhichen Wang[1] **Mario Bergés**[1] **Burcu Akinci**[1]

[1]Department of Civil and Environmental Engineering, Carnegie Mellon University, United States

zhichenw@andrew.cmu.edu, marioberges@cmu.edu, bakinci@cmu.edu

Abstract -

Building energy model (BEM) creation based on building information models (BIM) can save model re-creation time during building design phase. However, current BIM-to-BEM transformation is at the model level so the BEM is re-generated every time when the change happens to the BIM. Since design changes happen frequently and the generated BEM needs fine-tuning, such model regeneration is still time-consuming. Mapping rules between BIM and BEM are needed to achieve component level transformation, so that only the corresponding part in BEM instead of whole BEM are updated when changes happen to the BIM. These mapping rules can be defined explicitly using model transformation languages. However, these rule-based transformation methods have limitations in scalability. To solve this issue, this study proposes a pre-trained language model (PLM) based method to construct the mapping relationships between BIM and different types of BEM at the metamodel level. In summary, we formulate the BIM-BEM mapping as a machine translation task and solve it using PLM. For evaluation we collected and generated 35 pairs of BIM and BEM metamodels, and these metamodels are preprocessed into formatted texts that are readable by PLM. The 82% matching accuracy is achieved by proposed method which is higher than the 61% accuracy achieved by a baseline model in previous work. This paper shows the potential to utilize PLMs to facilitate the BIM to BEM transformation from BIM to a varying type of BEMs at the metamodel level. Future work will be focus on the realizing instance-level mapping and transformation.

Keywords -

Model transformation, Building information model, Building energy model, Interoperability, Natural language processing, Pre-trained language model

1 introduction

According to the report by International Energy Agency[1], the building sector accounted for 30% of global energy consumption and 27% of total energy sector emissions in 2021. Building energy models (BEMs) are utilized to predict energy consumption of the building and to improve its energy performance. However, traditional way to create BEM has some limitations: firstly, BEM modelers need to manually transform or rebuild the building geometry [2]. Secondly, the architects and BEM modelers are relatively separate in traditional BEM creation, which causes change conflicts [3, 4] and the discrepancy between architectural and thermal views [5, 6]. Therefore, building information model (BIM) based BEM modelling is proposed by prior studies since it can help automate the BEM modelling process so that the cost and re-modelling time can be saved [7, 8, 9], for example, Bazjanac [10] identified the potential time savings of 75% for creating the geometry of small and medium buildings through the appropriate application of automated processes.

Existing studies related to BIM-to-BEM transformation [11, 9, 12, 13, 5, 14] mainly focus on how to extract information from source BIM model and convert them into the format or description required by target BEM model. This BIM-to-BEM transformation process usually happens at the model level by generating the new BEM model based on the information from the corresponding BIM model. However, considering the fact that design changes frequently happen during the design phase, such model-level transformation means that the BEM is regenerated when the designer wants to update BEM based on the updated BIM, which results in the risk of losing previous BEM fine-tuning effort and expert knowledge (Seen Figure 1). Therefore, the mapping relationship at the component level should be constructed between BIM and BEM to realize the updating of necessary part of BEM instead of regenerating the whole BEM model. BIM-to-BEM transformation and mapping rules in current studies [15, 16, 17] rely on the manual definition with expert knowledge, but there are two main limitations. Firstly, existing methods require the explicit definition of rules between each class definition in BIM and BEM, and hence the number of rules needed to be defined is enormous due to large amounts of classes defined in BIM and BEM. For example, there more than 770 entities in IFC schema [18] and more than 500 classes in Modelica buildings library [19]. Second, different types of BEM, such as EnergyPlus and Modelica, are used in practice, and the transformation rules explicitly defined are specific to a BEM, so the scalability of these methods can be limited.

To overcome these issues, the research described in this paper utilizes an example-based method to learn transformation rules from pairs of BIM and BEM. Generally speaking, the model transformation process happens at two levels: instance and metamodel. The transformation

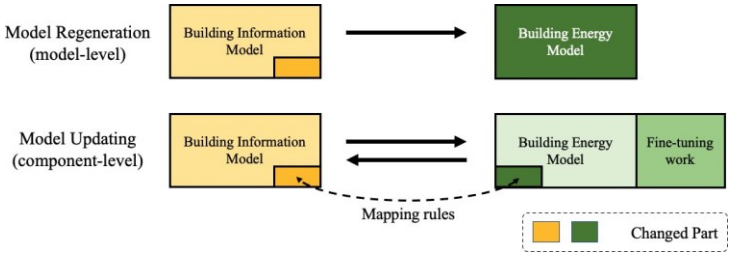

Figure 1. Model transformation at model level and component level

at the instance level focuses on the individual instantiated components while the one at the metamodel level focuses on the transformation between classes defined in the BIM and BEM models. Since the metamodels serve as the basis for instances, this paper will focus on transformations at the metamodel level. The proposed method regards the model transformation at the metamodel level as a text-to-text task. Therefore, a pretrained large language model, T5 (Text-to-text transfer transformer), is adopted and fine-tuned to assess the applicability of large language models in streamlining model transformation. The input is the metamodel of IFC4 files of building systems in JSON format, while the output is the corresponding metamodels of simulation models, including Modelica models, EnergyPlus models and CFD models, in JSON format. In the implementation, both the pretrained T5 model and baseline model from a prior study [20] are trained and evaluated based on the dataset containing 47 pairs of BIM and BEM metamodels with more than 2500 tokens. The proposed method achieves an accuracy of 84% in the token matching task on the test dataset, which is higher than the 61% accuarcy achieved by the baseline model. This highlights the applicability for large language models in helping with the model transformation and improving the scalability of model transformation between BIMs and various types of BEMs at the metamodel level.

2 Related work

We review the related work in two sub-fields: (1) BIM and BEM interoperability; (2) Model transformation in software engineering.

2.1 BIM and BEM interoperability

Based on the coupling relationships between building design tools and building simulation tools, current BIM-BEM interoperability approaches can be categorized into three types: centralized, distributed, and combined [21, 22]. Centralized BIM-BEM interoperability approaches utilize a central database, a file format or a schema such as BIM and then share it with other simulation tools. For example, OpenStudio[23] uses gbXML as the central schema, and Simergy [24] uses IFC as the central data schema for data exchange and model interoperability. Distributed approaches couple pairs of design tool and simulation tools through a middleware specific to them so that a point-to-point network is created between different design tools and simulation tools. The common example of using the distributed approach is to import the geometry model from design tools such as Revit or SketchUp to simulation tools such as EnergyPlus through a middlware. The combined approach usually packages design tools and simulation tools together so the models created by these tools communicate with each other internally, such as IESVE[25]. Compared with the other two types, centralized approaches enable high levels of customization, and usability of the central model can vary depending on the linked simulation model tools[22, 26]. Because of these advavntages, existing studies have largely focused on centralized interoperability approaches [15, 27, 28]. Therefore, in this paper we focus on the centralized methods for BIM and BEM interoperability, and leverage the most popular BIM representation, namely IFC [29].

Recent studies related to transforming BIMs to different types of BEMs are shown in the Table 1. The column of “Unique Requirement” shows the necessary information or data that are required by the transformation from BIM to each type of BEM. Meanwhile, items listed in this column are not shared by all other types of BEM due to the differences between the simulation engine or assumptions adopted by each type of BEM. For instance, the CFD model requires the mesh to enable the model creation and calculation while others do not [12, 13]. The Modelica, as the object-oriented simulation, requires the topology between instantiated components[11, 9]. The reviewed literature shows that the proposed BIM-to-BEM methods in current studies are limited to specific type of BEM models. However, practical application facts that various types of BEMs are widely adopted means that the BIM to BEM transformation method should have enough scalability to facilitate the transformation process for lower cost [30, 31]. Therefore, the low scalability is one common gap existing in the current BIM-to-BEM area. [32]

Table 1. BIM-to-BEM transformation literature summary

Type	Input	Output	Unique Information Requirement	Ref.
Object-oriented Model	IFC	Modelica	Topology;	[11, 9, 33, 34, 32]
Computational Fluid Dynamics	IFC	OpenFoam	Space boundary; Mesh; Geometry representation;	[12, 13]
Equation-based Model	IFC; gbXML; IDD;	EnergyPlus	HVAC system specification; Geometry representation; Load information;	[34, 5, 30, 14, 35]

2.2 Model transformation

Model transformation originates from the software engineering discipline[36], referring to the process of converting or mapping a model from one representation to another. The model is defined as the abstraction of a system or environment, and it can be a program code, UML model, or data schema [37]. The main intended applications of model transformation include (1) Model mapping or synchronizing at the same level or different levels of abstraction [38]; (2) Model evolution or refactoring [39]. The model transformation usually consists of three components: source model, target model and transformation rule [36, 32, 40]. To enable automatic model transformation, several model transformation languages are developed to define the transformation rules. Existing model transformation languages (MTLs) can be categorized into declarative MTL [41, 42], imperative MTL [43], and hybrid MTL[44, 45].

Model transformation languages explicitly define the transformation rules for each comparable pair of classes between source model and target model. However, considering the number of classes and entities in BIM and different types of BEMs, explicit definition brings high cost and limits in scalability [20, 46].

A previous study [47] shows that it is easier for experts to show transformation examples than to express complete and consistent transformation rules, so model transformation by example (MTBE) has since been more thoroughly explored. MTBE approaches can be categorized into search-based methods and machine learning based methods. The search-based methods regard the MTBE as an optimization problem and try to find the optimal one in provided mapped pairs [48, 49]. The machine learning based methods utilize machine learning models to learn the mapping rules based on provided mapped pairs [50, 20, 51].

The search-based MTBE approaches have limitations in requiring the mapping traces or defined transformation rules between example pairs, while the machine learning based MTBE shows potential to save cost and improve scalability without need for mapping traces.

In summary, two identified issues need to be mitigated for automated updating of BEMs. The first one is that there is a need to automate the definition of transformation rules between BIM and BEM, as this process can be time-consuming and costly due to the large number of entities and classes defined in IFC and the different BEM models. The second issue is that these transformation rules are specific to the different BEMs adopted in practice, including Modelica, CFD and EnergyPlus, requiring customization for each case.

In the next section, we introduce the machine learning based approache that we propose to learn the embedding mapping rules between BIM and different types of BEM at the metamodel level, so that explicit definition of transformation rules can be avoided and the scalability of the model transformation methods can be improved.

3 Method

This paper focuses on the model transformation at the metamodel level. The definition of metamodel and instances in this paper are the class defined in the schema and the entities instantiated from the defined classes respectively. Instead of directly using the entire schema containing all classes, we adopted portions of the schema (e.g. IFC4 schema [18]) or the class definition (e.g. Modelica Buildings Library[19]) as the metamodels. Metamodels of BIM and BEM are highly structured with defined classes and references among classes, so both of them can also be converted into structured text. Therefore, given the structured and textualizable BIM and BEM metamodels as input and output respectively, the model transformation process between BIM metamodel and BEM metamodel can be regarded as the text-to-text translation task, which is a typical natural language processing (NLP) task.

Considering the significant progress achieved in pretrained language models (PLMs) in recent years, in this paper we adopt a pretrained PLM, namely T5 (Text-to-Text Transfer Transformer) [52], as the base model to realize the transformation between metamodels of BIM and BEM. The T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks

with the idea of reframing all NLP tasks into a unified text-to-text-format, within which the input and output are always text strings [52]. Examples of tasks are machine translation, document summarization, question answering, and sentiment analysis.

The overview of the T5-based method for model transformation can be seen in Fig.2. The input of the proposed method is the metamodel of IFC, while the output is the metamodels of different types of BEMs. All these metamodels are created by following the standard schema or class definitions that are made publicly available, and they are represented in abstract syntax trees (AST) in the JSON format. The AST of metamodels follows the binary root-children structure which is proven to be effective for code translation by Chen et al[53]. The preprocessing block converts the JSON format AST of BIM metamodel into text, which is then sent to the T5 model. After the text generation, the post-processing block converts the generated text into JSON format AST of the output BEM metamodel. The pretrained T5 model is fine-tuned based on a dataset containing pairs of IFC and BEM metamodels, so that the implicit rules are learnt.

4 Implementation and results

To implement our proposed method we first need to obtain a robust set of samples to provide as input and output to our framework. Specifically, we need a collection of metamodel pairs (BIM as input and corresponding BEMs as output). Since, to the best of our knowledge, this dataset of metamodels does not exist (or at least not publicly), we resorted to creating the dataset ourselves following standard schemas and online documentation for the model class families we used. In particular, we created 44 metamodel pairs, all stemming from model instances representing a specific building system in a commercial building at Anonymous Location. The BIM metamodels in the implementation are IFC metamodels, while the BEM metamodels are metamodels of Modelica, CFD and Energyplus models.

As mentioned in Section 6, the metamodels of BIM and BEMs are converted into text files. The metamodels are initially represented using UML because it is suitable for visualization and manual operations. Subsequently, these metamodels are manually converted into an AST format and realized as JSON files for machine readability. Finally, we used Python to process these JSON-formatted metamodels into text files for tokenization. Figure 3 shows this conversion process using an example pair of IfcBoiler class and Boiler class in EngergyPlus. The AST of metamodels are created based on the metamodels visualized in the UML format, then the JSON formatted AST of metamodels are converted into plain text following given rules. Following above procedure, we created 44 pairs of BIM and BEM metamodels (Accessed through this link) divided them with the ratio of 70%, 15%, and 15%. The resulting training dataset contains 30 model pairs with around 1500 tokens, while both the validation and test dataset contain 7 model pairs with more than 650 tokens. As for the implementation of the pretrained language model, the T5-small model from HuggingFace transformer API [54] is adopted as the base model, then it is fine-tuned based on the collected metamodel pairs. The model training process is implemented under Google Colaboratory environment with 100 training epochs. Meanwhile, as a baseline, we benchmark against a LSTM neural network model from a previous study [20]. The baseline model also comprises the encoder-decoder architecture, but the encoder and decoder are implemented with a single-layer LSTM network [20]. A summary of the implementation configuration is shown in Table 2.

Table 2. Implementation configuration

Item	Value
Adopted model	T5-small (60 million parameters)
Baseline	LSTM encoder-decoder model [20]
Number of epochs	100
Batch size	4
Learning rate	0.002
Train dataset	1500 tokens, 30 model pairs
Test dataset	660 tokens, 7 model pairs
Validation dataset	680 tokens, 7 model pairs

The implementation results are evaluated by the accuracy of token matching on the test dataset. Given the input BIM metamodel and target BEM model type, the fine-tuned pretrained large language model generates the corresponding metamodel of BEM, then the accuracy is calculated by:

$$Acc = \frac{N_{matched}}{N_{tot\ al}}$$

where $N_{matched}$ is the number of matched token pairs, N_{total} is total the number of token pairs. As shown in the Table 3, when the pretrained T5-small model is trained after 100 epochs, the accuracy of token matching on the test dataset is around 82%, while the baseline model achieved 61% accuracy. Figure 5 shows examples of token matching results. The tokens highligted in bold are mismatched pairs. The mismatched pairs can be due to the type of attribute value or the name of attributes belonging to the given class.

5 Discussion

As shown in the previous section, our T5 pretrained model achieved an 82% token matching accuracy in the test set, which is higher than 61% accuracy achieved by the baseline model from [20]. This result points to the promise

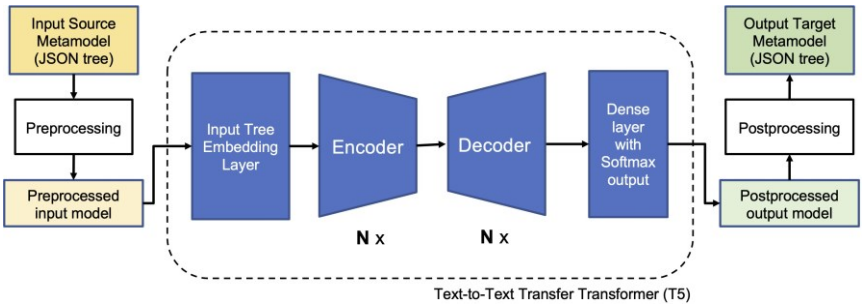

Figure 2. Proposed example-based model transformation method

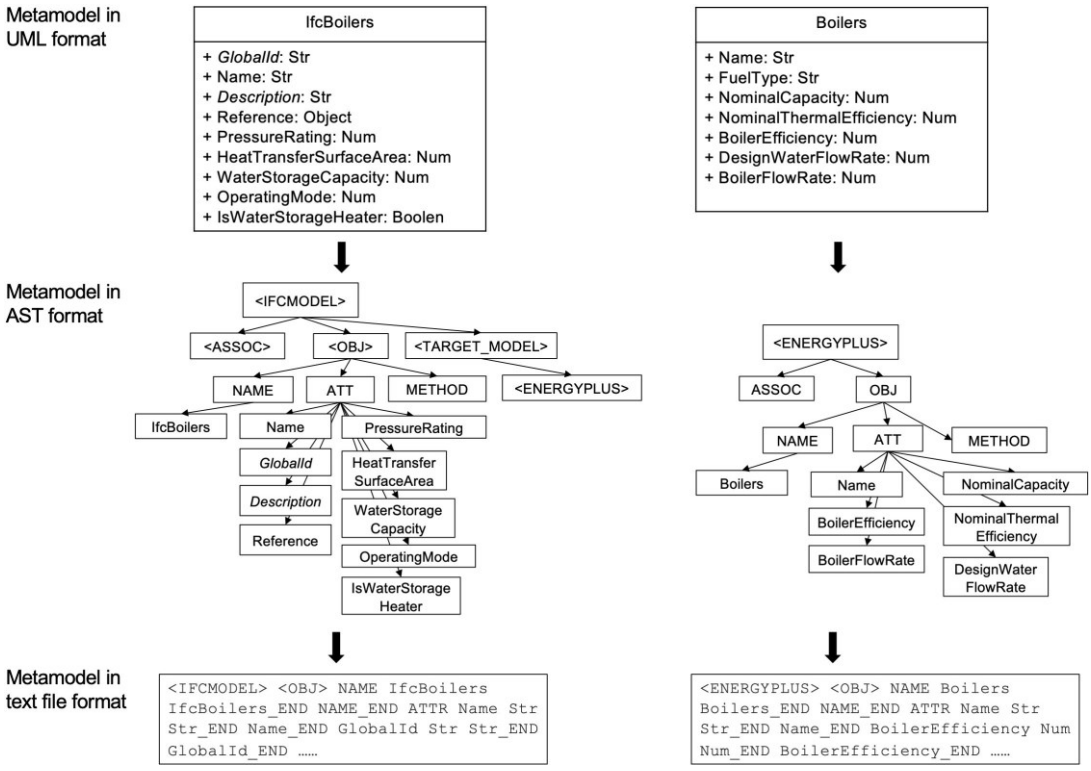

Figure 3. Workflow of converting the metamodels to text files

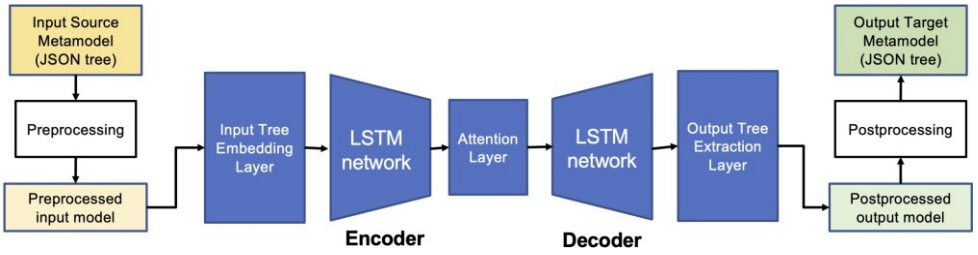

Figure 4. The workflow of LSTM based model transformation method from [20] as the baseline

Example 1	Ground Truth	<CFDMODEL> <OBJ> NAME CFDModel CFDModel_END NAME_END ATTR Volume Num Num_END Volume_END ThermalCapacity Num Num_END …… <unk>ASSOC> Composition Composition_END RoomGeometry RoomGeometry_END Wall Wall_END <unk>ASSOC>_END <CFDMODEL>_END
	Prediction	<CFDMODEL> <OBJ> NAME CFDModel CFDModel_END NAME_END ATTR Volume Num Num_END Volume_END ThermalCapacity Num Num_END …… <ASSOC> Composition Composition_END RoomGeometry RoomGeometry_END Wall Wall_END <ASSOC>_END <CFDMODEL>_END
Example 2	Ground Truth	<ENERGYPLUSMODEL> <OBJ> NAME ChillerHeater ChillerHeater_END NAME_END ATTR Name Str Str_END Name_END NominalCoolingCapacity **Real Real_END** ……
	Prediction	<ENERGYPLUSMODEL> <OBJ> NAME ChillerHeater ChillerHeater_END NAME_END ATTR Name Str Str_END Name_END NominalCoolingCapacity **Num Num_END** ……
Example 3	Ground Truth	<MODELICA> <OBJ> NAME …… ATT **res2** Real Real_END **res2_END flowCharacteristics3** Real Real_END **flowCharacteristics3_END res1** Real Real_END **res1_END** fraK ……
	Prediction	<MODELICA> <OBJ> NAME …… ATT **flowCharacteristics3** Real Real_END **flowCharacteristics3_END res1** Real Real_END **res1_END res2** Real Real_END **res2_END** fraK ……

Figure 5. Examples of token matching results

Table 3. Evaluation result on test dataset

#Epoch	Metric	Proposed	Baseline
30	Accuracy of tocken matching	0.21	0.22
100	Accuracy of tocken matching	0.82	0.61

of using model-based MTBE methodology to automate model transformation between BIM and BEM. In particular, it shows that we may not need domain-specific expert knowledge to define every transformation rule among entities and classes in the metamodels of BIM and BEMs. Concurrently, the incorporation of metamodels from three types of BEM within the training and testing datasets highlight the capability of the trained algorithm to fulfill the practical application demand of transforming the BIM into multiple types of BEM.

That said, the results also highlight that there are shortcomings to this method. Though we did not conduct experiments to fully characterize the failure casses, we can hypothesize about the possible reasons for the mismatched pairs of tokens highlightened in Figure 5. For example, as the examples in the figure show, the errors could be attributed to the characteristics of the training dataset and the representation format of metamodels. Firstly, for the mismatched pair of "eal" and "Num" in Example 2, these tokens stand for the data type of the attribute in those classes. In the training dataset, the occurence frequency "Num" surpasses 80%, while the "Real" accounts for less than 10% of the total number of tokens representing attribute data type. As a result, the model tends to predict the "Num" with higher probability. To mitigate this issue, diversifying the data sources can be explored to prevent the dominance of a specific data value in the dataset. Secondly, for the mismatched pairs of "res2", "flowCharacteristics3" and "res1" in Example 3, all of them are attributes of class Valves.ThreeWayTable in Modelica Buildings library. Lastly, as mentioned in Sections 6 and 4, the tree structured metamodel is represented using a sequential structure in plain text (AST). However, in the model prediction results, the generated "flowCharacteristics3" appears at the position of "res2" in the ground-truth. Since token matching is performed based on sequential positions, this pair is considered mismatched. One possible solution for this issue is to improve the representation format of the tree structure of metamodels in text file to avoid the impact of altered sequential positions of predicted tokens.

One significant difficulty met in the implementation of the proposed method is the acquisition and collection of data. There is a lack of open-source repositories providing metamodels of different types of BEMs, so the metamodels in the dataset were manually constructed by the authors, adhering to the online standard data schema and official documents. This limited dataset potentially exerts an impact on the performance of the fine-tuned large language model.

A further limitation of the PLM-based approach for BIM-to-BEM transformation pertains to the intricacies of instance-level transformation and mapping. This process necessitates the intricate alignment of specific attribute values of instances in BIM and BEM. However, these attribute values may exhibit randomness or be heavily influenced by contextual factors, leading to a scenario where the dataset collected for model fine-tuning does not cover all potential attribute values. For example, when considering a BIM-BEM pair that includes multiple room instances, these room instances vary in attributes such as their names, volumes, locations, and topological relationships with other instances. To address the challenge of instance-level transformation, one viable strategy involves the integration of constraints and rules that are specifically tailored to the attribute values. This approach would facilitate the transformation of instances with attribute values, from one instance from BIM to the other instance from BEM, after the instance class have been initially filtered through the metamodel transformation results derived from the PLM-based method.

6 Conclusion

Existing methods for model transformation between BIM and BEM are circumscribed in terms of cost and scalability. To mitigate these issues, this paper starts from the model transformation process at the metamodel level, and conceptualizes it as a text-to-text translation task. Subsequently, the pretrained large language model based MTBE approach is proposed to actualize the automated BIM to BEM model transformation at the metamodel level. To best of our knowledge, this paper is the first to apply the large language model based methods in the research area of BIM to BEM transformation. The implementation results envince the feasibility of the proposed method for BIM to BEM transformation. The contribution of this paper resides in (1) highlighting the applicability for large language models in helping with the model transformation and (2) improving the scalability by enabling the model transformation between BIM and various types of BEMs. To mitigate the issues of mismatched pairs observed the implementation result, the future work will focus on diversifying the data sources and improving the representation of metamodels in plain text. Meanwhile, the PLM-based method proposed in this paper is primarily concentrated on transformations at the metamodel level. Nonetheless, it encounters limitations when applied to transformations at the instance level, due to the necessity of converting specific attribute values for individual instances. Consequently, future work will be investigating methodologies for the incorporation of constraints and rules. This is aimed at facilitating the conversion of attribute values pertinent to instance-level transformation, thereby potentially enhancing the efficacy and applicability of the PLM-based approach in more complex, attribute-specific scenarios within the BIM to BEM transformation at all levels.

References

[1] International Energy Agency. Iea building energy report, 2022. URL www.iea.org/reports/buildings.

[2] Georgios N. Lilis, Georgios I. Giannakis, and Dimitrios V. Rovas. Automatic generation of second-level space boundary topology from IFC geometry inputs. *Automation in Construction*, 76:108–124, April 2017. ISSN 09265805. doi:10.1016/j.autcon.2016.08.044.

[3] Vladimir Bazjanac, Tobias Maile, Cody Rose, James T O'Donnell, Elmer Morrissey, and Benjamin R Welle. AN ASSESSMENT OF THE USE OF BUILING ENERGY PERFORMANCE SIMULATION IN EARLY DESIGN. In *Proceedings of Building Simulation*, Sydney, Australia, 2011.

[4] Benjamin Welle, John Haymaker, and Zack Rogers. ThermalOpt: A methodology for automated BIM-based multidisciplinary thermal simulation for use in optimization environments. *Building Simulation*, 4(4):293–313, December 2011. ISSN 1996-3599, 1996-8744. doi:10.1007/s12273-011-0052-5.

[5] Robert J Hitchcock, Justin Wong, and Hitchcock Consulting. TRANSFORMING IFC ARCHITECTURAL VIEW BIMS FOR ENERGY SIMULATION: 201.

[6] Vladimir Bazjanac. IFC BIM-based methodology for semi-automated building energy performance simulation. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), Berkeley, CA (United States), 2008.

[7] Mohamed H. Elnabawi. Building Information Modeling-Based Building Energy Modeling: Investigation of Interoperability and Simulation Results. *Frontiers in Built Environment*, 6:573971, December 2020. ISSN 2297-3362. doi:10.3389/fbuil.2020.573971.

[8] Gabriela Bastos Porsani, Kattalin Del Valle de Lersundi, Ana Sa´nchez-Ostiz Gutie´rrez, and Carlos Ferna´ndez Bandera. Interoperability between Building Information Modelling (BIM) and Building Energy Model (BEM). *Applied Sciences*, 11(5):2167, March 2021. ISSN 2076-3417. doi:10.3390/app11052167.

[9] WoonSeong Jeong and Kee Kim. A Performance Evaluation of the BIM-Based Object-Oriented Physical Modeling Technique for Building Thermal Simulations: A Comparative Case Study. *Sustainability*, 8(7):648, July 2016. ISSN 2071-1050. doi:10.3390/su8070648.

[10] Vladimir Bazjanac. SPACE BOUNDARY REQUIREMENTS FOR MODELING OF BUILDING GEOMETRY FOR ENERGY AND OTHER PERFORMANCE SIMULATION.

[11] Ando Andriamamonjy, Dirk Saelens, and Ralf Klein. An automated IFC-based workflow for building energy performance simulation with Modelica. *Automation in Construction*, 91:166–181, July 2018. ISSN 09265805. doi:10.1016/j.autcon.2018.03.019.

[12] Minhyung Lee, Gwanyong Park, Hyangin Jang, and Changmin Kim. Development of Building CFD Model Design Process Based on BIM. *Applied Sciences*, 11(3):1252, January 2021. ISSN 2076-3417. doi:10.3390/app11031252.

[13] Eric Fichter, Veronika Richter, Je´roˆme Frisch, and Christoph van Treeck. Automatic generation of second level space boundary geometry from IFC models. In *2021 Building Simulation Conference*, September 2021. doi:10.26868/25222708.2021.30156.

[14] Oliver Spielhaupter. *BIM to BEM Transformation Workflows: A Case Study Comparing Different IFC-Based Approaches*. PhD thesis.

[15] Issa J. Ramaji, John I. Messner, and Ehsan Mostavi. IFC-Based BIM-to-BEM Model Transformation. *Journal of Computing in Civil Engineering*, 34(3):04020005, May 2020. ISSN 0887-3801, 1943-5487. doi:10.1061/(ASCE)CP.1943-5487.0000880.

[16] Jong Bum Kim, WoonSeong Jeong, Mark J Clayton, Jeff S Haberl, and Wei Yan. Developing a physical bim library for building thermal energy simulation. *Automation in construction*, 50:16–28, 2015.

[17] Yikun Yang, Yiqun Pan, Fei Zeng, Ziran Lin, and Chenyu Li. A gbXML Reconstruction Workflow and Tool Development to Improve the Geometric Interoperability between BIM and BEM. *Buildings*, 12(2):221, February 2022. ISSN 2075-5309. doi:10.3390/buildings12020221.

[18] buildingSmart. Industrial foundation classes (ifc) 4.0, 2020. URL https://standards.buildingsmart.org/IFC/RELEASE/IFC4_1/FINAL/HTML/link/annex-e.htm.

[19] Lawrence Berkeley National Laboratory. Modelica buildings library, 2023. URL https://simulationresearch.lbl.gov/modelica/.

[20] Loli Burguen˜o, Jordi Cabot, Shuai Li, and Se´bastien Ge´rard. A generic LSTM neural network architecture to infer heterogeneous model transformations. *Software and Systems Modeling*, 21(1):139–156, February 2022. ISSN 1619-1366, 1619-1374. doi:10.1007/s10270-021-00893-y.

[21] Goran Sibenik and Iva Kovacic. Assessment of model-based data exchange between architectural design and structural analysis. *Journal of Building Engineering*, 32:101589, November 2020. ISSN 23527102. doi:10.1016/j.jobe.2020.101589.

[22] Kristoffer Negendahl. Building performance simulation in the early design stage: An introduction to integrated dynamic models. *Automation in Construction*, 54:39–53, June 2015. ISSN 09265805. doi:10.1016/j.autcon.2015.03.002.

[23] NREL U.S. Department of Energy. Openstudio, July 10, 2023. URL https://openstudio.net/.

[24] Digital Alchemy. Lbnl simergy, lawrence berkeley natl. lab, 2023. URL https://d-alchemy.com/products/simergy.

[25] Integrated Environmental Solutions. Integrated environmental solutions, iesve, July 10, 2023. URL https://www.iesve.com/.

[26] Hang Li and Jiansong Zhang. Interoperability between BIM and BEM Using IFC. In *Computing in Civil Engineering 2021*, pages 630–637, Orlando, Florida, May 2022. American Society of Civil Engineers. ISBN 978-0-7844-8389-3. doi:10.1061/9780784483893.078.

[27] Cody M. Rose and Vladimir Bazjanac. An algorithm to generate space boundaries for building energy simulation. *Engineering with Computers*, 31(2):271–280, April 2015. ISSN 0177-0667, 1435-5663. doi:10.1007/s00366-013-0347-5.

[28] Huaquan Ying and Sanghoon Lee. An algorithm to facet curved walls in IFC BIM for building energy analysis. *Automation in Construction*, 103:80–103, July 2019. ISSN 09265805. doi:10.1016/j.autcon.2019.03.004.

[29] Shu Tang, Dennis R Shelden, Charles M Eastman, Pardis Pishdad-Bozorgi, and Xinghua Gao. Bim assisted building automation system information exchange using bacnet and ifc. *Automation in Construction*, 110:103049, 2020.

[30] Hang Li and Jiansong Zhang. Improving IFC-Based Interoperability between BIM and BEM Using Invariant Signatures of HVAC Objects. *Journal of Computing in Civil Engineering*, 37(2):04022059, March 2023. ISSN 0887-3801, 1943-5487. doi:10.1061/(ASCE)CP.1943-5487.0001063.

[31] Hang Li, Jiansong Zhang, Soowon Chang, and Anthony Sparkling. BIM-based object mapping using invariant signatures of AEC objects. *Automation in Construction*, 145:104616, January 2023. ISSN 09265805. doi:10.1016/j.autcon.2022.104616.

[32] Jun Cao. *SimModel Transformation Middleware for Modelica-based Building Energy Modeling and Simulation*. PhD thesis.

[33] Matthis Thorade, Jo¨rg Ra¨dler, Peter Remmen, Tobias Maile, Reinhard Wimmer, Jun Cao, Moritz Lauster, Christoph Nytsch-Geusen, Dirk Mu¨ller, and Christoph van Treeck. An Open Toolchain for Generating Modelica Code from Building Information Models. In *The 11th International Modelica Conference*, pages 383–391, September 2015. doi:10.3384/ecp15118383.

[34] James O'Donnell, Richard See, Cody Rose, Tobias Maile, Vladimir Bazjanac, and Phil Haves. SIMMODEL: A DOMAIN DATA MODEL FOR WHOLE BUILDING ENERGY SIMULATION.

[35] Veronika Richter, Eric Fichter, Maximilian Azendorf, Je´roˆme Frisch, and Christoph van Treeck. Algorithms for Overcoming Geometric and Semantic Errors in the Generation of EnergyPlus Input Files based on IFC Space Boundaries.

[36] Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches. *IBM systems journal*, 45(3): 621–645, 2006.

[37] Perdita Stevens. A landscape of bidirectional model transformations. *Generative and Transformational Techniques in Software Engineering II: International Summer School, GTTSE 2007, Braga, Portugal, July 2-7, 2007. Revised Papers*, pages 408–424, 2008.

[38] Igor Ivkovic and Kostas Kontogiannis. Tracing evolution changes of software artifacts through model synchronization. In *20th IEEE International Conference on Software Maintenance, 2004. Proceedings.*, pages 252–261. IEEE, 2004.

[39] Gerson Sunye´, Damien Pollet, Yves Le Traon, and Jean-Marc Je´ze´quel. Refactoring uml models. In *International Conference on the Unified Modeling Language*, pages 134–148. Springer, 2001.

[40] Ethan Post, Kevin Dinkel, Erich Lee, Bjorn Cole, Hongman Kim, and Bassem Nairouz. Cloud-based orchestration of a model-based power and data analysis toolchain. In *2016 IEEE Aerospace Conference*, pages 1–12, Big Sky, MT, USA, March 2016. IEEE. ISBN 978-1-4673-7676-1. doi:10.1109/AERO.2016.7500648.

[41] Yue Cao, Yusheng Liu, Hongri Fan, and Bo Fan. SysML-based uniform behavior modeling and automated mapping of design and simulation model for complex mechatronics. *Computer-Aided Design*, 45(3):764–776, March 2013. ISSN 00104485. doi:10.1016/j.cad.2012.05.001.

[42] Andy Schu¨rr and Felix Klar. 15 Years of Triple Graph Grammars. In Hartmut Ehrig, Reiko Heckel, Grzegorz Rozenberg, and Gabriele Taentzer, editors, *Graph Transformations*, volume 5214, pages 411–425. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-87404-1 978-3-540-87405-8. doi:10.1007/978-3-540-87405-8_28.

[43] Joel Greenyer and Ekkart Kindler. Comparing relational model transformation technologies: Implementing Query/View/Transformation with Triple Graph Grammars. *Software & Systems Modeling*, 9(1):21–46, January 2010. ISSN 1619-1366, 1619-1374. doi:10.1007/s10270-009-0121-8.

[44] Marcel van Amstel, Steven Bosems, Ivan Kurtev, and Lu´ıs Ferreira Pires. Performance in Model Transformations: Experiments with ATL and QVT. In Jordi Cabot and Eelco Visser, editors, *Theory and Practice of Model Transformations*, volume 6707, pages 198–212. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21731-9 978-3-642-21732-6. doi:10.1007/978-3-642-21732-6_14.

[45] Dimitrios S. Kolovos, Richard F. Paige, and Fiona A. C. Polack. The Epsilon Transformation Language. In Antonio Vallecillo, Jeff Gray, and Alfonso Pierantonio, editors, *Theory and Practice of Model Transformations*, volume 5063, pages 46–60. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-69926-2 978-3-540-69927-9. doi:10.1007/978-3-540-69927-9_4.

[46] K. Lano, S. Kolahdouz-Rahimi, and S. Fang. Model Transformation Development Using Automated Requirements Analysis, Metamodel Matching, and Transformation by Example. *ACM Transactions on Software Engineering and Methodology*, 31(2):1–71, April 2022. ISSN 1049-331X, 1557-7392. doi:10.1145/3471907.

[47] U. Behrens, M. Flasinski, L. Hagge, J. Jurek, and K. Ohrenberg. Recent developments of the ZEUS expert system ZEX. *IEEE Transactions on Nuclear Science*, 43(1):65, February 1996. ISSN 0018-9499, 1558-1578. doi:10.1109/23.486006.

[48] Islem Baki and Houari Sahraoui. Multi-Step Learning and Adaptive Search for Learning Complex Model Transformations from Examples. *ACM Transactions on Software Engineering and Methodology*, 25(3):1–37, August 2016. ISSN 1049-331X, 1557-7392. doi:10.1145/2904904.

[49] Zolta´n Balogh and Da´niel Varro´. Model transformation by example using inductive logic programming. *Software & Systems Modeling*, 8(3):347–364, July 2009. ISSN 1619-1366, 1619-1374. doi:10.1007/s10270-008-0092-1.

[50] Loli Burgueno, Jordi Cabot, and Sebastien Gerard. An LSTM-Based Neural Network Architecture for Model Transformations. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 294–299, Munich, Germany, September 2019. IEEE. ISBN 978-1-72812-536-7. doi:10.1109/MODELS.2019.00013.

[51] Paulius Danenas and Tomas Skersys. Exploring Natural Language Processing in Model-To-Model Transformations. *IEEE Access*, 10:116942–116958, 2022. ISSN 2169-3536. doi:10.1109/ACCESS.2022.3219455.

[52] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1): 5485–5551, 2020.

[53] Xinyun Chen, Chang Liu, and Dawn Song. Tree-to-tree Neural Networks for Program Translation, October 2018.

[54] HuggingFace. Huggingface transformer api, 2023. URL https://huggingface.co/docs/transformers/index.