A Large Language Model as an Assistant for Software-Independent Structural Analytical Model Review and Editing

Justin S. Lee[1] and Ghang Lee[1,2*]

[1]Department of Architecture and Architectural Engineering, Yonsei University, Seoul, Republic of Korea
[2]Institute for Advanced Studies, Technical University of Munich, Munich, Germany
*Corresponding author
justinlee@yonsei.ac.kr, glee@yonsei.ac.kr

Abstract

While structural analysis software facilitates the modeling of complex structures, it requires significant effort to master these tools. With advancements in large language models (LLMs), this study explores the potential of leveraging an LLM as a software-independent assistant for reviewing and editing structural analytical models. The proposed framework was tested using GPT-4o and an analytical model from Midas Gen, focusing on tasks such as model information extraction and editing. The results demonstrated the framework's potential, achieving an accuracy of 91% when a system prompt with model data ontology information was used, compared to a 20% accuracy without the use of any prompt engineering.

Keywords –

Analytical Models; Structural Models; LLM; System Prompt; Code Interpreter

1 Introduction

Computer based structural analysis has become an indispensable part of the structural engineering profession. The emergence of structural analysis software has enabled engineering designs of much higher complexity and greater scale. However, even with the evolution of structural analysis software over the years, the general modeling approach has not evolved dramatically – they still rely heavily on heuristics and trial-and-errors which makes the structural design optimization a human-centric process [10]. In this task, the extent of the engineer's ability to efficiently control the analytical model constructed within the analysis software becomes an important factor. Especially for engineers who perform designs for multiple industries or regions, they may be required to obtain proficiency in multiple structural analysis software which is no small feat. Furthermore, even if adequate software proficiency is reached, engineers are usually limited to the native features of the software. This lack of flexibility in analytical model control leads to tasks with high interaction cost, especially when dealing with complex models.

Large language models (LLMs) have made great strides in recent years. With carefully designed prompts, LLMs demonstrate great performance on textual classification, information extraction, and summarization [11]. Considering the building blocks of any structural analytical model are simple textual and numeric arrays, an LLM based user interaction method for analytical model control can potentially provide much more freedom in communicating the engineer's intent to the structural analysis software. In addition to this already powerful textual processing abilities, multimodal LLMs allow for amalgamation of diverse set of data types [9].

This study aims to lay the foundation for developing a software-independent structural analytical model review and editing method using an LLM for future research endeavors. To perform a structural analysis, a structural model and its properties must be prepared, reviewed, and adjusted for the targeted analysis. The process of establishing a structural model frame can be broken down as shown in Figure 1. This study focuses particularly on the last two steps: review and editing steps. For the review step, the ability to select specific groups of elements and extract various information about them is evaluated. For the editing step, the capability to adjust element attributes, such as section types or boundary conditions, for a specified group of elements is assessed.

Selecting a specific group of elements within the modeling software, or filtering, is an essential feature in performing review and editing steps of any structural analytical model. Filtering is generally done with natively offered basic filter options. There are software packages that allow users to build custom filters such as SAP2000 and its advanced filter features using a

structural query language (SQL); however, while such approach widens the types of filters that can be used, it still does not offer enough flexibility to cover all possible scenarios and are often time consuming to build, test, and deploy when needs arise. For example, if a user needs to filter beams that are not directly connected to any columns to select all sub-beams or joists, such filtering would generally not be covered by native software filters. This element selection via filtering must be followed by the ability to apply edits to the element attributes, if necessary, completing the review and editing process.

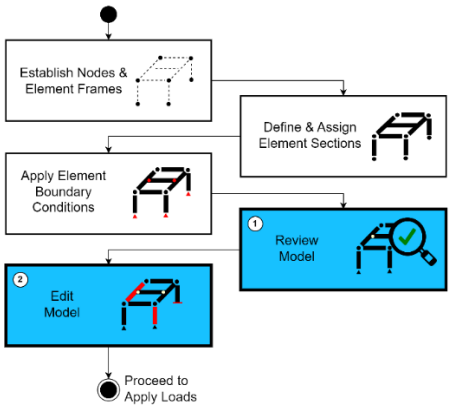

Figure 1. Review and editing steps of structural analytical modeling process

For testing, the geometric and element feature data of an analytical model from the most widely used structural analysis software in South Korea, Midas Gen, was uploaded to ChatGPT-4o along with a system prompt and model data ontology information to enhance its performance. Then, a series of tasks were given as user prompts where corresponding results were evaluated.

2 Related Studies

This section of the study consists of two parts: The first part will review literature on LLM applications in architectural design process where the primary focus will be LLM augmentation on design software. The second part will be an overview of artificial intelligence (AI) applications in structural engineering where model generation and design optimization will be discussed.

2.1 LLM Augmentation on Design Software

Natural language (NL) based user expressions allow for much more complex forms of queries to be used in software control. One application strategy is to extract user's intentions via LLM and map specific functions or features of the design software. For instance, an NL query to search for a specific element or additional information about the element from a building information modeling (BIM) entity, such as a Revit model, has been proposed [11]. In this literature, the NL query is first processed by the LLM to map its intent to the corresponding parameters and value dictionaries. Then, the appropriate search functions are called to interact with the application programming interface (API). However, this approach limits its focus on element searching and does not account for design generation or information management.

Jang et al. expands the prompt-based user interface concept to handle Revit element manipulation with their proposed framework Natural-language-based Architectural Detailing through Interaction with AI (NADIA) [4]. In this framework, the user request via the task prompt is first categorized based on its task type and relevant component type. Then, the necessary BIM authoring is executed to generate, modify, retrieve, or delete the relevant Revit object. This functionality more closely resembles the analytical model control which will be discussed in this study; however, the main difference will be that the geometric data and some of the element feature data will be entirely transferred to LLM for this study where analytical model data will be manipulated directly by LLM without any reliance on existing software functions or features.

2.2 AI Application in Structural Engineering

Design optimization is expectedly a popular topic of interest in many recent studies involving AI and structural engineering. As per most engineering design problems, structural design problems involve a large set of input variables and also many output variables to be optimized. Structural optimization problems can be categorized into cost, performance, environmental impact, and multi-objective optimization (MOO) problems [8]. For example, reinforced concrete section design is an MOO problem that is an excellent candidate for an optimization method using an artificial neural network (ANN). ANN can be used to explore various pareto curves to flexibly optimize what would traditionally be a computationally expensive problem with much less effort [2]. However, design process optimization, including user-software interaction, unfortunately falls outside of the purview of this popular view of engineering optimization.

Although it is not an optimization method, generative AI presents new opportunities for structural topology generation. AI models can be used to extract patterns from a large design dataset and reproduce resembling structural layouts accordingly [7]. Generative Adversarial Network (GAN) can be used to generate column and beam models based on 2D images [6], furthermore, physical rules can be implemented alongside GAN to enhance the quality of the generated structural topology [1]. Although these novel strategies

contribute to enhancing the qualities of the AI generated analytical models, the need for structural engineers to closely review and edit these models still remains.

3 Proposed Method

3.1 Analytical Model Data

For most commercial structural analysis software, such as Midas Gen or ETABS, analytical model data is typically stored in a series of 2D arrays or tables. Analytical model data composition for Midas Gen is summarized in Table 1. These tables reference each other – for example, the element table does not contain the coordinates of these elements explicitly, but rather they specify the two nodes each element is connected to. A part of the "Elements" table from Midas Gen is shown in Table 2.

Table 1. Midas Gen model data tables

Table Name	Contents
Elements	Element Number / Material / Section / Orientation / Node Connectivity
Nodes	Node Numbers / Coordinates
Materials	Material Number / Name / Type / Sub-type
Supports	Node Assignment / Translation & Rotation Release
Sections	Section Number / Name / Shape / Sizes

Table 2. "Elements" table from Midas Gen

Element ID	Material	Section	Beta Angle	Node 1	Node 2
1	1001	101	90	1	10
2	1001	103	0	1	11
3	1001	101	90	2	11
4	1001	103	0	3	11
5	1001	101	0	3	12

Compared to typical architectural models, structural analytical models are much simpler and hold vastly less information as they are based on directional stick-and-node graphs. This simpler and smaller model data can be uploaded to ChatGPT-4o for more direct control of the model data without relying on any built-in software functionalities for model manipulations. To ensure that ChatGPT-4o properly understands the data structure, an ontology representation file was created with Protégé as shown in Figure 2.

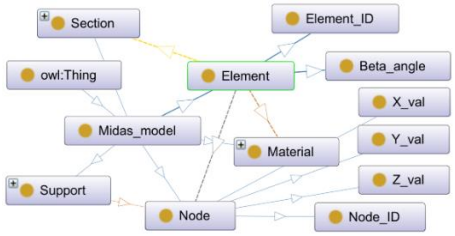

Figure 2. Midas Gen model data ontology

3.2 Data Processing

When tabular data, such as CSV files, is uploaded to ChatGPT-4o, it is stored in its memory as DataFrames which is a table-like data format provided by the Python library pandas. When the user prompts inquire about the information stored in the DataFrames, ChatGPT-4o then writes and executes its own Python codes to extract and edit the data (Figure 3). This functionality of ChatGPT-4o is called Code Interpreter. These codes are then immediately available to the user with the response. An example of the code generated by ChatGPT-4o is shown in Figure 4 in pseudo-code for clarity.

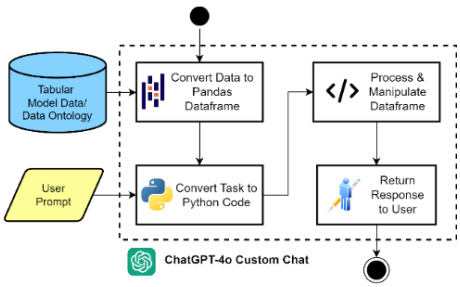

Figure 3. Tabular data processing of ChatGPT-4o

```
Given 'element_data' and 'node_data'
1  Set 'first_floor_z' = 0.0m
2  Extract all column elements from 'element_data'
   into 'columns'
3  Initialize 'first_floor_column_count' = 0
4  For each 'row' in 'columns':
5      Retrieve coordinates (x1, y1, z1) of the first
       node from 'node_data'
6      Retrieve coordinates (x2, y2, z2) of the second
       node from 'node_data'
7      If either z1 or z2 is equal to 'first_floor_z':
8          Increment 'first_floor_column_count' by 1
9  Return 'first_floor_column_count'
Output 9
```

Figure 4. ChatGPT-4o generated code for the task 'How many columns are on the first floor?'

Because the analytical model review and edit assistant proposed in this study relies on ChatGPT-4o to correctly interpret the user tasks and successfully extract and manipulate the relevant information, the system heavily depends on ChatGPT-4o's ability to write effective Python codes.

3.3 System Prompt

The system prompt was developed through a trial-and-error approach, iteratively refining its content based on errors until ChatGPT-4o could visualize the analytical model successfully. After several iterations, the following three main components were selected to be included in the textual part of the system prompt: general role, geometric context, and element definitions as shown in Table 3. The general role explains the expected role of ChatGPT-4o followed by the descriptions of the CSV model data files and the data ontology file. The geometric context explains the contextual meaning of the coordinate system, for example, z = 0.0 means ground level. Finally, the element definitions describe the definitions of various structural components in terms of their nodal coordinates. The element definitions component of the system prompt is important because the user is expected to refer to the elements by the element types, such as beams or columns; however, ChatGPT-4o, in its native state, was not able to identify these element types solely based on the model data.

Followed by the textual portion of the system prompt, the data ontology file was attached to the system prompt of ChatGPT-4o; however, the CSV model data files were not uploaded with the system prompt for two reasons: first, the proposed framework needed to work with any model data, not just one set of CSV files. Second, ChatGPT-4o was unable to keep and recall all the information properly when textual system prompt, the ontology file, and the CSV files were all uploaded within the system prompt at once. In other words, ChatGPT-4o was unable to process multiple instructions and files all under one request as LLMs generally struggle with queries with multiple objectives and perform better when interrelated tasks are provided separately [3].

3.4 Framework Assembly

The analytical model review and edit framework is assembled using the custom chat feature of ChatGPT: The system prompt is first inserted in the instructions window, followed by uploading the data ontology file (.rdf format) prepared using Protégé. This custom chat is saved and initiated as an instance, expecting to receive the analytical model CSV files extracted from Midas Gen. For this study, the CSV files were directly uploaded to ChatGPT-4o's web interface. Once CSV files are processed, it is then ready to take on the user tasks. The overall flow of this framework is shown in Figure 5.

Table 3. The textual portion of the system prompt

Category	Contents
General Role	• You are a virtual assistant to help review and edit an analytical structural model. The model data is stored in several CSV files. These files reference each other. You can combine these files referencing an ontology (.rdf format) file which I will provide also. Let me give you the ontology file first.
Geometric Context	• Ground level is first floor (z = 0.0m). • The second floor is one story above the first floor. • Roof level is the highest level.
Element Definitions	• Columns are vertical elements (change in ONLY 'z' coordinates). • Beams are horizontal elements (no change in 'z' coordinates between nodes). • Braces are diagonal elements (change in ('x' or 'y') coordinates AND 'z' coordinates). • 'Beta_angle' means local orientation, not global.

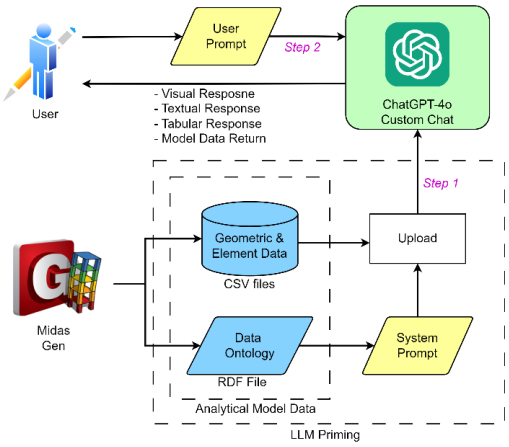

Figure 5. Structural analytical model review and edit framework

4 Validation

To evaluate the analytical model review and edit framework, this study focused on its ability to accurately execute user queries. A diverse set of tasks, including visualization, review, and editing were designed to

evaluate the key capabilities of the framework. The task set and the descriptions are summarized in Table 4.

4.1 Test Subject

A modified version of the American Society of Civil Engineers (ASCE) benchmark structure from the Earthquake Engineering Research Laboratory at the University of British Columbia (UBC) [5] was modeled in Midas Gen. Minor modifications were introduced to this model to add more geometric complexity and incorporate additional section types.

Table 4. Validation Task Set

Task Type	Task Sub-Type	Description
Visualization	Create 3D view	Render a 3D view of the model.
	Color by element type	Show beams, columns, braces and support nodes in assorted colors.
Extraction for Review	Single table reference	Inquire about information accessible from a single table.
	Multiple table reference	Inquire about information that requires accessing multiple tables that reference each other.
Extraction and Editing	Based on prompts	Edit model data based on user prompts.
	Based on images	Edit model data based on user prompts and image files.

4.2 Evaluation Task Categories

4.2.1 Visualization Tasks

The purpose of the visualization task category is to verify that the custom chat initialization has been correctly performed. Although these tasks do not simulate any potential user inquiries, for the validation of the study, it is a key step to ensure that the framework is able to assemble the model data correctly. The geometric assembly is validated by the 3D render task and the element classification is validated by the element color assignment task (Figure 6).

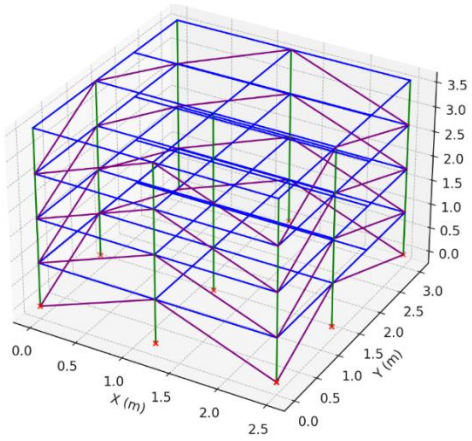

Figure 6. A successful visualization task result

4.2.2 Extraction for Review Tasks

The extraction for review task category is to simulate the user inquiries to extract model information for the purpose of reviewing various model attributes, such as element quantities, section types, material types or connection types. This task category includes basic filtering tasks to ensure that the LLM based approach can effectively replace the basic filters offered by most modeling software as well as more challenging tasks such as boundary condition-based filtering and filtering based on neighboring element types such as adjacency with columns.

For the single table reference tasks, the requested information is directly accessible from one of the DataFrames, while for the multiple table reference tasks, ChatGPT-4o must utilize the data ontology to reference multiple DataFrames to process and generate the appropriate response. An example of a multiple table reference task would be an element length calculation task where the element node numbers refer to the nodes table where the coordinates are stored. The coordinates are then used to calculate the Euclidean distance between the two nodes to evaluate the element length. An image file containing grid label assignment is also uploaded at this stage of testing.

4.2.3 Extraction and Editing Tasks

The extraction and editing task category consists of two important steps: firstly, the framework needs to understand the user request to extract element attributes and make changes to the model data stored in its memory, then secondly, it needs to return CSV files back to the user in a format that is consistent with the original model data. The second step is crucial to allow the CSV files to be loaded back to Midas Gen for an uninterrupted workflow.

The extraction and editing task based on images is to simulate a scenario where structural elements need to be

updated in accordance with drawings, either from architectural or structural design changes. For this task, an image (.png format) file containing a plan view with revised section type labels is uploaded along with the user prompt.

5 Results

A total of 11 tasks were tested 5 times each. In addition, to see the effects of the system prompt containing the textual instructions and the data ontology file, the same set of tests were repeated without any initialization (ChatGPT-4o in its vanilla state) and with the system prompt. In total, 110 task trials were conducted for this study. The test results are summarized in Figure 7 and detailed results along with the task prompts are shown in Table 5.

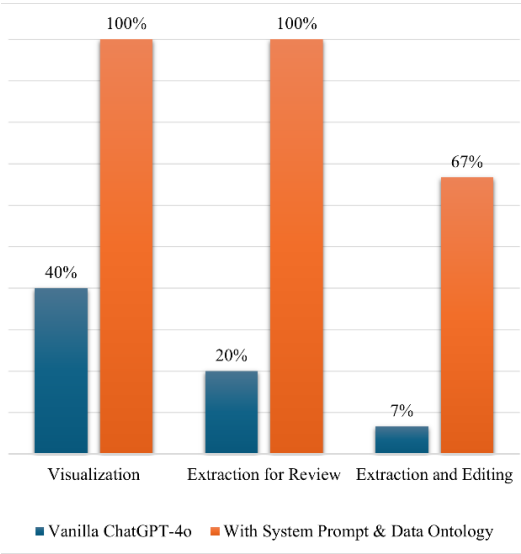

Figure 7 Test results summary

5.1 Results by Task Categories

5.1.1 Visualization Tasks

ChatGPT-4o was able to successfully perform the 3D visualization task 4 out of 5 times even without any assistance from the system prompt; however, it was not able to classify different element types, such as beams, columns, or braces. Although ChatGPT-4o has access to general definitions of what each of those element types are, it was not able to semantically connect the dots between element definitions and geometric orientations in the cartesian coordinate system.

When initialized with the system prompt that contains detailed element definitions in the context of the model geometry, both 3D visualization and color coding by element type tasks were performed perfectly.

5.1.2 Extraction for Review Tasks

The overall success rate of the extraction for review tasks without the system prompt was 20%. A notable result was when asked to check if all beams that are not directly connected to columns also pin connected, the correct response (‘yes’) was given 3 out of 5 times even without the system prompt. To ensure the successful responses were not ‘false positive’ answers, an additional inquiry was made to identify the relevant element numbers for confirmation. With the system prompt, all the tasks were performed perfectly for all trials.

5.1.3 Extraction and Editing Tasks

ChatGPT-4o was able to return the edited CSV files in the correct column arrangements (to allow for direct import back to Midas GEN) even without the system prompt; however, it was able to perform the requested edits correctly for only 1 out of the 15 trials. The most common errors were edits being applied to too many elements around the correct target elements. Prompt-based editing tasks were perfectly performed with the system prompt. Image-based editing tasks could not be performed successfully even with the system prompt as the target elements were always misidentified.

6 Discussion

Recent LLM driven BIM software control studies have generally relied on the software’s natively built-in functionalities for model control. This study demonstrates that leveraging ChatGPT-4o’s code interpreter feature to more directly control analytical model data is a feasible strategy, with one important caveat: the framework requires an elaborate system prompt, including detailed textual instructions along with data ontology information, to function effectively.

6.1 Limitations

While this study provides new insights into a prompt-based analytical model review and editing framework that is free of any predefined software functionalities, there are limitations that must be addressed.

The proposed framework was evaluated using the ASCE benchmark model, which is significantly less complex than most real-world engineering examples. This limitation restricts the generalizability of the results, as the structural simplicity of the ASCE model may not adequately represent the challenges encountered in practical engineering scenarios. In addition, the validation process was conducted with a relatively small and narrowly defined set of tasks, further constrained by a limited number of repetitions.

Table 5. Task user prompts and test results

Task Type	Task Sub-type	User Prompt	Success/Fail: Vanilla ChatGPT-4o	Success/Fail: With System Prompt & Data Ontology
Visualization	3D View of Model	a. Create a 3D visualization of analytical model.	4/5	5/5
		b. Show support nodes in red, and color beams, columns, and braces differently.	0/5	5/5
Extraction for Review	Single table reference	a. How many columns are there on the first floor?	1/5	5/5
		b. What are the element numbers of beams on the ‘roof level’ along grid X1?	0/5	5/5
		c. Give me the element numbers of beams that are not connected to columns.	0/5	5/5
	Multiple table reference	a. Give me the element lengths of all the beams on the third floor.	1/5	5/5
		b. Give me the element numbers and section names of beams that are on the third floor and beam end released	1/5	5/5
		c. Are all the beams that are not directly connected to columns also beam-end-released?	3/5	5/5
Extraction and Editing	Based on prompts	a. Replace all beam sections on the roof level to 'section 101'. Apply this change and return the element CSV in its original column format.	0/5	5/5
		b. Remove all beams with beam end released on the second level. Apply this change and return the element CSV in its original column format.	1/5	5/5
	Based on images	a. Apply beam section changes as per the uploaded PNG on the ‘roof level’. Select the section number according to the section name in the PNG file. Apply this change and return the element CSV in its original column format.	0/5	0/5
		Overall Success Rate	20%	91%

The framework's performance depends entirely on ChatGPT-4o's ability to generate effective Python code for manipulating model data. Also, heavy reliance on an LLM can lead to unreliable reproducibility. As demonstrated by the test results, variations were observed in the responses generated, even when identical input conditions were provided.

Failure to perform image-based tasks is a crucial drawback of this framework. Drawing based structural analytical model revision is a common and time-consuming task, and it would be an invaluable functionality to be implemented to this framework.

6.2 Future Work

Further testing with structural models that more closely resemble the size and complexity of real-world engineering examples with greater number of repetitions will be conducted. A more unified pipeline will also be developed for the proposed framework for proper testing and further development.

For the upcoming research efforts, the effects of the data ontology representation independently from the system prompt will be explored to better quantify its impact on the performance of the proposed framework.

Image-based task performance must be addressed by exploring new strategies such as few-shot learning or more refined system prompt designs.

7 Conclusion

This study explored the potential of LLMs as analytical model review and editing assistants for structural analysis. By using an LLM-driven approach, it demonstrates new ways to improve user interaction beyond traditional tools like rule-based filters. The results showed the framework’s potential, achieving an accuracy of 91% when a system prompt with model data ontology information was used, compared to a 20% accuracy without any prompt engineering.

Key limitations include evaluation with the relatively simple ASCE benchmark model with limited validation task set and repetitions, an overreliance on ChatGPT-4o's capability to produce effective Python code and difficulties in managing image-based model edit tasks. Future work will focus on more rigorous validation and developing a unified pipeline for seamless integration with industry-standard tools. The image-based task performance will also be addressed by exploring few-shot learning and more refined system prompts.

8 Acknowledgements

This work is supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant RS-2021-KA163269 and RS-2024-00407028) and the Hans Fischer Senior Fellowship program at the Technical University of Munich - Institute for Advanced Studies (TUM-IAS) in Germany.

References

[1] Fu B., Wang W., and Gao Y. Physical rule-guided generative adversarial network for automated structural layout design of steel frame-brace structures. Journal of Building Engineering, 86:108943, 2024.

[2] Hong W.-K. and Nguyen D.H. Pareto frontier for steel-reinforced concrete beam developed based on ANN-based Hong-Lagrange algorithm. Journal of Asian Architecture and Building Engineering, 22(6):3535–3551, 2023.

[3] Hu H., Yu S., Chen P., and Ponti E.M. Fine-tuning large language models with sequential instructions. On-line: https://doi.org/10.48550/arXiv.2403.07794, Accessed: 20/11/2024.

[4] Jang S., Lee G., Oh J., Lee J., and Koo B. Automated detailing of exterior walls using NADIA: Natural-language-based architectural detailing through interaction with AI. Advanced Engineering Informatics, 61:102532, 2024.

[5] Johnson E.A., Lam H.F., Katafygiotis L.S., and Beck J.L. Phase I IASC-ASCE structural health monitoring benchmark problem using simulated data. Journal of Engineering Mechanics, 130(1):3–15, 2004.

[6] Kösencig̃ K.Ö., Okuyucu E.B., and Balaban Ö. Structural plan schema generation through generative adversarial networks. Nexus Network Journal, 26(3):409–427, 2024.

[7] Málaga-Chuquitaype C. Machine learning in structural design: An opinionated review. Frontiers in Built Environment, 8:815717, 2022.

[8] Mei L. and Wang Q. Structural optimization in civil engineering: A literature review. Buildings, 11(2):66, 2021.

[9] Qi S., Cao Z., Rao J., Wang L., Xiao J., and Wang X. What is the limitation of multimodal LLMs? A deeper look into multimodal LLMs through prompt probing. Information Processing & Management, 60:103510, 2023.

[10] Zhao P., Liao W., Huang Y., and Lu X. Intelligent design of shear wall layout based on attention-Engineering Structures, 274:115170, 2023.

[11] Zheng J. and Fischer M. Dynamic prompt-based virtual assistant framework for BIM information search. Automation in Construction, 155:105067, 2023.