Exploring Large Language Model-driven Automated Generation of BIM Parametric Designs

Xiaoqian Peng*

Guangzhou Vocational and Technical University of Science and Technology, Guangzhou 510000, China

*Corresponding email: 479777828@qq.com

Abstract

Building Information Modeling (BIM) underpins digital management across the architecture, engineering, and construction (AEC) sector, yet parametric model creation remains a largely manual process. Practitioners must define component parameters, configure family properties, and specify constraint relationships within commercial BIM platforms - a workflow that is time-consuming and demands specialized expertise. This paper explores a method that uses large language models (LLMs) to automate the generation of BIM parametric designs from natural language descriptions. The proposed pipeline consists of three modules: a semantic parsing module that extracts structured design parameters from text prompts via prompt engineering, a parameter mapping module that translates these parameters into Industry Foundation Classes (IFC) entity attributes through a curated mapping dictionary, and a model generation module that produces standard IFC model files using IfcOpenShell with integrated geometric consistency checks. We evaluate the method on 30 natural language inputs covering three structural systems with varying complexity. For simple descriptions, parameter extraction accuracy reaches 87%, and the model generation success rate is 78%, reducing the requirement-to-model cycle by roughly 60% compared to traditional scripting approaches. The main sources of error are JavaScript Object Notation (JSON) format inconsistencies in large language model (LLM) output and elevation conflicts between components. We discuss limitations related to output determinism, code compliance, and domain coverage, and propose retrieval-augmented generation, domain fine-tuning, and human-Artificial Intelligence (AI) collaborative verification as directions for future improvement.

Keywords

Building information modeling, Large language models, Parametric design, Automated generation, Industry Foundation Classes, Natural language

Introduction

Building Information Modeling (BIM) has become the standard infrastructure for digital management in the architecture, engineering, and construction (AEC) industry. BIM consolidates geometric information, physical properties, component relationships, and lifecycle data into an integrated model, supporting management from schematic design through construction, operation, and demolition [1]. Governments worldwide have mandated BIM adoption for public projects, and the technology is now deeply embedded in contractual workflows, quantity surveying, and facility management. Despite this broad adoption, a fundamental bottleneck persists: Model generation itself remains a manual craft. Professionals must individually define component parameters, adjust family properties, and configure constraints within software such as Revit and ArchiCAD. For a single standard floor plan, this process can take hours; for complex buildings with varied structural systems, the effort scales considerably. The bottleneck is not in storage or visualization but in the translation of design intent into structured parametric data.

Scripting tools such as Dynamo (for Revit) and Grasshopper (for Rhino) offer partial automation by allowing users to encode parametric rules in visual programming or Python scripts. These tools reduce repetitive work but still require the operator to possess both programming skills and domain knowledge of BIM

Application Programming Interface (APIs). The learning curve is steep, and scripts are often project-specific, difficult to generalize, and costly to maintain across software version updates. A more accessible interface - one that allows designers to express requirements in plain language - would lower the barrier to parametric modeling and broaden its adoption.

Large language models (LLMs) present a compelling mechanism for bridging this gap. Built on the Transformer architecture, LLMs such as Generative Pre-trained Transformer 4 (GPT-4) and Large Language Model Meta AI (LLaMA) demonstrate strong capabilities in natural language understanding, code generation, and structured information extraction [2-4]. In the broader software engineering domain, LLMs have been shown to generate functional code from natural language specifications with reasonable accuracy, and structured output techniques have improved their reliability for producing JSON, Extensible Markup Language (XML), and other machine-readable formats. These capabilities suggest a natural fit for BIM parametric design, where the task can be framed as translating a textual description of design intent into a structured set of component parameters.

Several research threads have begun converging toward this vision. In structural engineering, Çalışkan showed that ChatGPT can generate meaningful calculation scripts from natural language requirements, though with limited code compliance and numerical precision [5]. In facility management, LLMs have been combined with BIM databases to build natural language query systems that allow non-expert users to retrieve building spatial information through dialogue. In generative design, Wang & Liao surveyed Generative Adversarial Networks (GANs) and diffusion models for floor plan generation, noting that the gap between generated floor plans and parametric BIM models remains open [6]. Kim et al. explored deep learning approaches for extracting constraint relationships from floor plan data to reverse-generate BIM component parameters [7]. Most recently, Du et al. proposed Text to Building Information Modeling (Text2BIM), a multi-agent framework that uses LLMs to generate three-dimensional (3D) building models from natural language, and Ko et al. introduced a conversational AI framework for parametric modeling in architectural design [8,9]. A systematic review by Gao et al. further confirmed that LLM adoption in construction is accelerating, though applications remain concentrated in information retrieval and documentation rather than model generation [10].

Despite these advances, a systematic exploration of LLM-driven BIM parametric design generation - from natural language input through IFC model output - has not yet been thoroughly documented. Key challenges include controlling output precision across varying description complexity, resolving semantic ambiguity in architectural terminology, ensuring the determinism of structured outputs, and validating generated models against building codes. This paper addresses these challenges by proposing a three-module pipeline, evaluating it on a curated set of test cases, and analyzing specific failure modes that constrain current performance.

In this paper, we explore an LLM-driven approach for automated generation of BIM parametric designs. We propose a pipeline consisting of three modules - semantic parsing, parameter mapping, and model generation - and evaluate it on 30 test cases covering three structural systems with description complexity ranging from simple to complex. The contributions of this work are threefold. (1) A concrete pipeline design that connects natural language input to standard IFC model output is proposed. (2) An empirical evaluation quantifying accuracy, success rate, and geometric correctness across complexity levels is conducted. (3) An analysis of failure modes that identifies concrete directions for improvement is provided.

Related work

BIM parametric modeling and its automation

Automated BIM parametric modeling has evolved through several technological phases. The earliest efforts relied on rule-based knowledge engineering: expert systems encoded architectural design codes as inference rules, enabling limited design automation within narrowly defined domains [11]. These systems were brittle - they could not generalize beyond their encoded rule sets and required substantial manual maintenance as codes changed.

The rise of machine learning brought data-driven approaches. Brown et al. proposed a method that used deep learning to extract spatial constraint relationships from existing floor plan layouts and then reverse-generate BIM component parameters [12]. This approach shifted the paradigm from rules to data, but it was limited by the availability and quality of training data and could not easily incorporate new design requirements expressed in natural language.

Visual programming environments - notably Dynamo for Revit and Grasshopper for Rhino - introduced parametric automation accessible to designers who could learn a visual scripting language. These tools have been widely adopted for generative design exploration, but they still require the operator to manually wire parameters and logic nodes. A recent review of generative and parametric design integration with BIM highlighted that the gap between conceptual design intent and executable parametric models remains a central barrier to wider adoption.

More recently, generative AI has opened new avenues. Wang & Liao reviewed generative models for architectural plan generation, showing that GANs and diffusion models can produce floor plans satisfying functional zoning requirements. However, the conversion from raster or vector floor plans into fully parametric BIM models with correct IFC entity relationships remains an unsolved problem. Du et al. proposed Text2BIM, a multi-agent LLM framework that generates 3D building models from natural language, demonstrating that LLMs can serve as coordinators across specialized modeling agents. While Text2BIM represents a significant step forward, it relies on complex multi-agent orchestration that introduces its own failure modes.

Large language models in engineering and construction

LLM applications in the AEC sector have expanded rapidly since 2023. Gao et al. conducted a systematic review of LLM use in smart construction, identifying documentation automation, safety monitoring, and code compliance checking as the most active application areas. In structural engineering, Çalışkan investigated ChatGPT-assisted structural design calculations, finding that LLMs can generate meaningful scripts but fall short on code compliance and numerical accuracy - a finding consistent with broader observations about LLM reliability in technical domains.

In facility management, LLMs are combined with BIM databases to build natural language query systems that allow non-expert users to retrieve building spatial information through dialogue. These applications leverage LLMs on the retrieval and interpretation side of BIM data but do not address the generative side - creating new BIM models from design descriptions.

Prompt engineering has emerged as a key technique for improving LLM output quality without fine-tuning. Structured output prompting, in which the model is instructed to produce responses in JSON or XML format, has been shown to improve consistency and downstream parseability. Few-shot prompting - providing domain-specific examples within the prompt - further improves accuracy for specialized tasks. These techniques form the basis of the semantic parsing module in our pipeline.

Retrieval-augmented generation (RAG), introduced by Lewis et al., addresses a fundamental limitation of LLMs: Their knowledge is frozen at the training cutoff date and cannot be easily updated [13]. RAG augments LLM responses with relevant documents retrieved from an external corpus at inference time. This approach is particularly relevant for BIM applications, where building codes, material specifications, and design standards are frequently updated and must be strictly observed.

Nevertheless, integrating LLMs with BIM still faces challenges in data interoperability, real-time validation, and handling of ambiguous design descriptions. Addressing these limitations will require domain-adaptive fine-tuning and tighter coupling between generative LLMs and rule-checking engines.

Method architecture

The proposed LLM-driven BIM parametric design generation method consists of three core modules that operate sequentially: semantic parsing, parameter mapping, and model generation. The pipeline takes a natural language description of design intent as input and produces a standard IFC format BIM model file as output. The overall architecture of the proposed pipeline is shown in Figure 1.

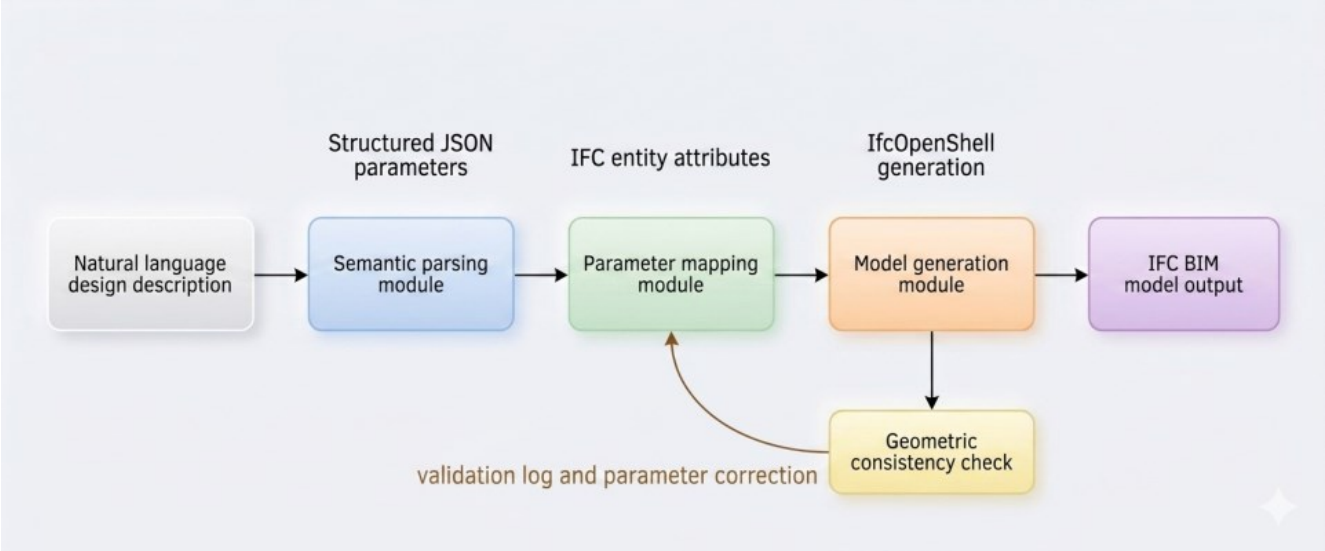

Figure 1. Architecture of the LLM-driven BIM parametric design generation pipeline.

Semantic parsing module

The semantic parsing module is responsible for converting free-form natural language into structured design parameters. Users describe their requirements in plain text - for example, "Generate a standard floor plan: bay width 8.4 m, depth 6.0 m, story height 3.6 m, concrete frame structure, Concrete strength grade 30 (C30) concrete grade." This description is fed to an LLM (GPT-4 or a comparable model) along with a carefully designed prompt template.

Prompt template design is central to this module. (1) It specifies the required output format (JSON), ensuring that the downstream parameter mapping module can reliably parse the result. (2) It defines the expected parameter categories (geometric dimensions, structural system, material specifications, story count, etc.), providing the model with a schema to follow. (3) It includes 2-3 domain-specific examples of input-output pairs, leveraging few-shot learning to improve output consistency for architectural terminology. The template also includes explicit instructions to avoid hallucinated parameters and to flag any input values that fall outside typical design ranges.

The structured JSON output from this module contains fields such as structural_system (e.g., "frame", "shear_wall", and "frame_shear_wall"), bay_width, depth, story_height, number_of_stories, material_grade, and load_description. Each field is typed (string, float, or integer), and optional fields are marked with null values when not specified in the input. This structured representation serves as the interface between the LLM and the downstream mapping and generation modules.

Parameter mapping module

The parameter mapping module translates the semantic parameters extracted in the previous step into IFC entity attributes. IFC is the international open standard (International Organization for Standardization (ISO) 16739-1:2024) for BIM data exchange, maintained by buildingSMART International. IFC defines a comprehensive property schema for building components, including geometric representations, material properties, spatial relationships, and type definitions. The standard is supported by all major BIM platforms and serves as the lingua franca for interoperability.

This module continuously maintains a parameter mapping dictionary - a structured lookup table that accurately maps natural language design concepts to their corresponding IFC representations. For example, "bay width" maps to the span attribute of Industry Foundation Class Beam (IfcBeam) and the placement coordinates of Industry Foundation Class Column (IfcColumn). "Story height" maps to the local placement Z-offset of Industry Foundation Class Slab (IfcSlab) and IfcBeam. "Concrete frame structure" triggers the creation of IfcColumn, IfcBeam, and IfcSlab entities with corresponding material properties in Ifc Mechanical Material Properties. "C30 concrete grade" maps to specific strength values in the material property set.

During mapping, the module performs several operations beyond simple lookup. Unit conversion is applied where necessary (e.g., converting millimeters in the IFC schema to the meters used in natural language input). Default values are filled for parameters not explicitly specified by the user, drawing from a database of common residential design defaults (e.g., slab thickness of 120 mm for residential floors, beam cross-section of 300 mm × 600 mm for typical frame structures). Constraint validation checks that the parameter set is internally consistent - for example, verifying that the bay width does not exceed the maximum span allowed by the selected structural system and material grade. When a constraint violation is detected, the module flags the issue and attempts to adjust parameters within a predefined tolerance range.

Model generation module

The model generation module produces a standard-format BIM model file from the mapped IFC parameter set. It uses IfcOpenShell, an open-source C++/Python library for IFC file creation and manipulation, as the primary generation engine. IfcOpenShell provides a programmatic APIs for creating IFC entities, setting properties, establishing spatial containment relationships, and writing the result to an .IFC file that can be opened in any IFC-compliant BIM viewer.

The generation process follows a predefined construction sequence. First, the project and site contexts are created (Industry Foundation Classes Project (IfcProject), Industry Foundation Classes Site (IfcSite)). Then, the building and story structures are established (Industry Foundation Classes Building (IfcBuilding), Industry Foundation Classes Building Storey (IfcBuildingStorey)). Next, structural components (columns, beams, slabs, walls) are placed at their computed coordinates. Finally, material properties and type definitions are attached. Each component is generated according to the parameter set from the mapping module, with geometric representations created using extruded area solids for linear elements and surface models for slabs.

After generation, the module runs a geometric consistency check. This check performs basic collision detection by comparing the bounding boxes of adjacent components and flagging overlaps that exceed a configurable tolerance threshold. The check also verifies that component elevations are consistent with the story structure - for example, ensuring that beam bottom elevations align with slab top elevations at each story level. Components that fail these checks are reported in a validation log, allowing the user to identify and correct issues before importing the model into a BIM platform.

Experiments and analysis

Experimental setup

We designed 30 natural language input examples targeting typical residential floor plans. The examples cover three structural systems: frame structure (10 examples), shear wall structure (10 examples), and frame-shear wall structure (10 examples). Description complexity is classified into three levels. The simple level includes only geometric parameters, such as bay width, depth, and story height. The moderate level includes geometric parameters plus structural system and material grade. The complex level includes geometric parameters, structural system, material specifications, and load descriptions. The distribution across complexity levels is roughly equal within each structural system category.

Three metrics are used to evaluate performance. Parameter extraction accuracy measures the match rate between the fields extracted by the semantic parsing module and the expected fields in the ground-truth parameter set. It is computed as the fraction of correctly extracted fields out of all expected fields for each example. Model generation success rate measures the proportion of examples for which the pipeline produces a valid, parseable IFC file without errors in the generation module. Geometric correctness measures the proportion of successfully generated models that pass the geometric consistency check (bounding-box collision detection and elevation alignment verification). The LLM backend is GPT-4 (GPT-4-turbo), accessed through the Open Artificial Intelligence (OpenAI) APIs with a temperature setting of 0.1 to reduce output randomness. Each input example is run three times, and the majority-vote output is used as the final result. The parameter mapping dictionary contains 45 entries covering common residential building components

(columns, beams, slabs, walls, foundations) and their IFC property mappings.

Results

Figure 2 summarizes the overall performance of the proposed pipeline across three evaluation metrics and three description-complexity levels. The comparison clearly shows that performance declines as the input description becomes more complex, especially for parameter extraction accuracy and model generation success rate.

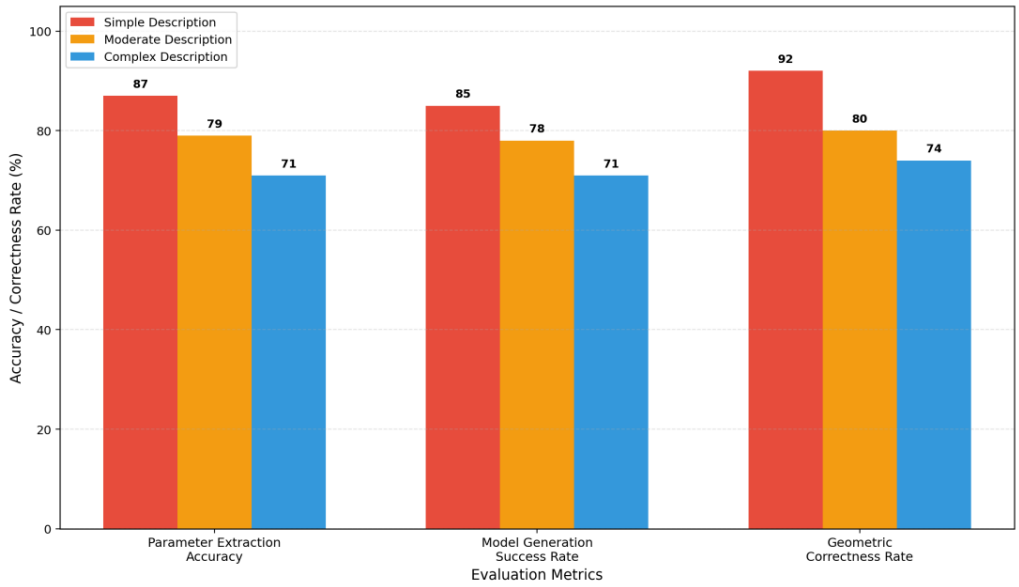

Figure 2. Comparison of evaluation metrics across description complexity levels.

For simple descriptions, parameter extraction accuracy reaches 87%, reflecting the LLM's ability to parse straightforward geometric specifications reliably. Accuracy decreases to 79% for moderate descriptions and 71% for complex descriptions. The main sources of degradation at higher complexity are ambiguity/polysemy, omission of standard parameters, and unit conversion errors, particularly when users omit units or mix measurement conventions.

The overall model generation success rate is 78%. This success rate, while moderate, supports further development. To examine the causes of unsuccessful generation, the observed failures were categorized according to the error types shown in Figure 3.

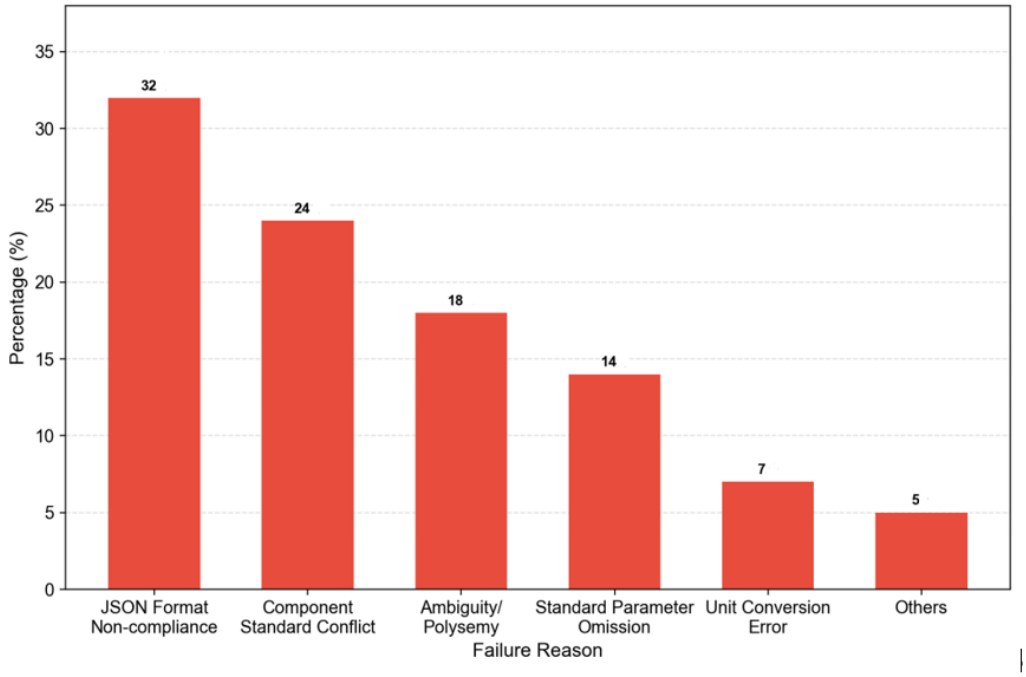

Figure 3. Distribution of model generation failure causes.

As shown in Figure 3, the largest failure category is JSON Format Non-compliance, accounting for 32% of failures. This category refers to cases in which the LLM output does not conform to the predefined JSON schema, such as nesting parameters under incorrect keys or using inconsistent field names. The remaining failures are associated with Component Standard Conflict (24%), Ambiguity/Polysemy (18%), Standard Parameter Omission (14%), Unit Conversion Error (7%), and Others (5%).

Geometric correctness among successfully generated models is 82%. The remaining 18% of models contain at least one elevation conflict or bounding-box overlap. Component elevation conflicts occur most frequently at beam-slab junctions in frame-shear wall structures. The interaction between beams and walls creates complex elevation relationships. These relationships are not fully captured by the current mapping rules.

Compared with traditional scripting-based modeling, such as Dynamo scripts for Revit, the proposed method reduces the requirement-to-model cycle by approximately 60%. A typical simple floor plan that takes an experienced Dynamo user 30-45 minutes to script can be generated from a natural language description in under five minutes using the proposed pipeline, including LLM inference time and post-generation validation. This speed improvement comes at a cost: Scripting-based models can achieve near-perfect geometric correctness, whereas the LLM-driven approach reaches an 82% geometric correctness rate. This efficiency-precision trade-off indicates that the method is most suitable for rapid prototyping and early-stage design exploration, where human verification remains necessary before engineering use.

Discussion

Method limitations

The experimental results reveal several limitations that constrain the practical applicability of the current approach.

(1) Output determinism

Even with a low temperature setting (0.1) and majority voting, the LLM occasionally produces different JSON structures across repeated runs of the same input. This non-determinism contributes to JSON Format Non-compliance and can cause intermittent failures in the parameter mapping module. While majority voting mitigates the issue, it triples the inference cost and does not eliminate occasional mode collapse where all three runs produce the same incorrect output.

(2) Domain coverage

The current parameter mapping dictionary covers common residential building components. Industrial buildings (factories, warehouses), infrastructure projects (bridges, tunnels), and special structural forms (long-span, high-rise, seismic-isolated) are not well supported. Extending coverage requires adding new mapping entries and corresponding IFC entity creation logic, which is labor-intensive and demands domain expertise for each new building typology.

(3) Code compliance

The method relies on prompt-supplied examples for understanding design requirements. However, it cannot internalize regulatory constraints from building codes. These constraints include fire safety separation distances, minimum structural member dimensions, or maximum allowable span-to-depth ratios. Generated models may satisfy geometric consistency checks while violating code provisions, creating compliance risks if models are used beyond exploratory design phases without human review.

Future directions

Three directions appear promising for addressing the limitations identified above.

First, retrieval-augmented generation (RAG) could substantially improve code compliance. Index relevant building code texts (e.g., the Chinese Code for Design of Concrete Structures GB 50010, the Code for Fire Protection Design of Buildings GB 50016) and make them accessible to the LLM at inference time. The semantic parsing module could then cross-reference generated parameters against regulatory requirements in real time. Lewis et al. demonstrated that RAG improves factual accuracy in knowledge-intensive tasks, and the construction domain - where code compliance is safety-critical - is a natural application area.

Second, domain-specific instruction fine-tuning could address the terminology and output stability issues. A BIM-specific instruction dataset, constructed from paired natural language descriptions and IFC parameter

sets, could be used to fine-tune an open-source LLM such as LLaMA. Fine-tuning would reduce the reliance on prompt engineering and improve output schema adherence. It would also enable the model to learn domain-specific conventions (e.g., default slab thicknesses, typical beam cross-sections) that are currently handled by the mapping dictionary.

Third, a human-AI collaborative verification workflow could combine the speed of automated generation with the reliability of human judgment. After model generation, a visual diff interface could present the generated model alongside the original natural language description. It could highlight parameter values that deviate from expectations, and flag components that failed geometric checks. The user could then confirm, adjust, or reject individual parameters before committing the model, creating a closed-loop workflow of AI generation followed by human verification. This hybrid approach is consistent with the broader trend toward human-in-the-loop AI systems in safety-critical engineering domains.

Conclusion

This paper presents a robust LLM-driven approach for automated generation of BIM parametric designs from natural language descriptions. The proposed pipeline consists of three modules - semantic parsing, parameter mapping, and model generation - that together translate a textual design specification into a standard IFC model file. Experiments are systematically conducted on 30 test cases. These cases comprehensively cover three structural systems (frame, shear wall, and frame-shear wall) at three complexity levels. The results clearly demonstrate that the method is fully feasible for simple to moderate design descriptions. Specifically, the parameter extraction accuracy reaches 87% for simple inputs. The overall model generation success rate is 78%. The method reduces the requirement-to-model cycle by approximately 60% compared to traditional scripting approaches, though at a cost in geometric correctness (82% vs. near-perfect for scripted models).

The analysis of failure modes identifies several primary sources of error. These include JSON Format Non-compliance in LLM output, Component Standard Conflict, Ambiguity/Polysemy, Standard Parameter Omission, Unit Conversion Error, and domain coverage gaps in the parameter mapping dictionary. These findings point to concrete improvement paths. First, retrieval-augmented generation can be applied for code compliance. Second, domain-specific fine-tuning helps improve output stability. Third, human-AI collaborative verification ensures safety assurance. As LLM capabilities continue to advance and BIM open standards mature, AI-driven parametric design automation has genuine potential for practical engineering adoption. This potential is particularly evident as a rapid prototyping tool in early-stage design. In such stages, speed matters more than absolute precision.

Funding

This work was supported by the 2025 Private Education Research Base Project of Guangdong Academy of Educational Sciences (Grant No. 2025JD07).

Acknowledgements

The author would like to show sincere thanks to those techniques who have contributed to this research.

Conflicts of Interest

The author declares no conflict of interest.

References

[1] Zou, Y., Kiviniemi, A., Jones, S. W. (2017) A review of risk management through BIM and BIM-related technologies. *Safety Science,* 97, 88-98.

[2] Zhang, Y., Liu, C., Liu, M., Liu, T., Lin, H., Huang, C.-B., Ning, L. (2024) Attention is all you need: Utilizing attention in AI-enabled drug discovery. *Briefings in Bioinformatics*, 25(1), bbad467.

[3] Sanderson, K. (2023) GPT-4 is here: what scientists think. *Nature*, 615(7954), 773.

[4] Aydın, Ö., Karaarslan, E., Erenay, F. S., & Bacanin, N. (2025) Generative ai in academic writing: A comparison of deepseek, qwen, chatgpt, gemini, llama, mistral, and gemma. *Turkish Journal of Engineering*, 10(2), 592-607.

[5] Çalışkan, E. B. (2025) Exploring possibilities and limits of ChatGPT: usage in building design studies. *Turkish Journal of Engineering*, 9 (3), 490-500.

[6] Wang, Y., Liao, W. (2026) Generative AI-driven architectural and structural engineering designs: a review. *Journal of Intelligent Construction*, 4, 9180126.

[7] Kim, S., Jeong, K., Hong, T., Lee, J., Lee, J. (2023) Deep learning-based automated generation of material data with object-space relationships for scan to BIM. *Journal of Management in Engineering*, 39(3), 04023004.

[8] Du, C., Esser, S., Nousias, S., Borrmann, A. (2026) Text2BIM: Generating building models using a large language model-based multiagent framework. *Journal of Computing in Civil Engineering*, 40(2), 04025142.

[9] Ko, J., Ajibefun, J., Yan, W. (2025) Generative AI-powered parametric modeling and BIM for architectural design and visualization. *Proceedings of the Design Society*, 5, 1943-1952.

[10] Gao, Y., Yiu, T. W., Shen, X., Tam, V. W. (2026) Large language models in smart construction: a systematic review of implementation strategies, applications and future directions. *Engineering, Construction and Architectural Management*, 33(15), 159-181.

[11] Fischer, M., Kunz, J. (2004) The scope and role of information technology in construction. *Proceedings-Japan Society of Civil Engineers*, 1-32.

[12] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Amodei, D. (2020) Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877-1901.

[13] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kiela, D. (2020) Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.