Proof-of-Concept Framework of Integrating AI-based LLM Reasoning into BIM Workflows for Design Automation

Jinwu Xiao[1], Cansu Coskun[1], Soowon Chang[1] *, Kyubyung Kang[1], Deniz Besiktepe[1]

[1]School of Construction Management Technology, Purdue University, United States

*** Corresponding Author**

xiao270@purdue.edu, ccoskun@purdue.edu, soowonchang@purdue.edu, kyukang@purdue.edu, denizb@purdue.edu

Abstract -

Building Information Modeling (BIM) has significantly advanced design efficiency and collaboration in the architecture, engineering, and construction (AEC) industry. However, current BIM workflows remain hindered by complex interfaces, repetitive tasks, and limited automation options, which result in a steep learning curve and labor-intensive design processes. To address the challenges, this research presents a proof-of-concept framework that explores the integration of Artificial Intelligence (AI) and Large Language Models (LLMs) with BIM environments to streamline design modifications through natural language interaction. Using a specialized FreeCAD integration layer, the framework demonstrates the feasibility of translating high-level user instructions into basic BIM operations. Initial testing shows this approach can successfully create simple parametric building elements while reducing token consumption and user effort compared to a baseline approach using generic tools. This feasibility study suggests potential for developing more intuitive BIM interfaces, though further research is needed to achieve robust domain knowledge integration and handle complex architectural relationships. The findings indicate promising directions for future development of LLM-driven BIM automation tools that could enhance accessibility for AEC professionals.

Keywords -

BIM Design Automation; Natural Language Processing; Computer-Aided Design; Human-Computer Interaction

1 Introduction

Building Information Modeling (BIM) has revolutionized the architecture, engineering, and construction (AEC) industry by providing comprehensive digital representations of the physical and functional characteristics of buildings [1]. Although BIM platforms have significantly improved design efficiency and collaboration, practitioners still face challenges such as complex interfaces, repetitive tasks, and redundant manual operations, which increases the need for automated workflows [2]. However, current BIM workflows lack user interaction and accessibility to automation. Traditional approaches to BIM interaction lack intuitive interfaces for design modifications and automation capabilities, leading to reduced efficiency and increased risk of errors during the design process [3].

Recent advances in Artificial Intelligence (AI) and Large Language Models(LLMs) have shown promising capabilities to understand natural language commands and execute complex computer operations [4]. The remarkable progress in AI has enabled architects to harness computational power and data-driven algorithms to generate and assess design options [5]. AI technologies in the design process have enabled architects to explore innovative and eco-friendly design solutions. Similarly, construction industry professionals have witnessed the evolution of digital data acquisition technologies, which have enhanced monitoring processes by delivering efficient and effective results [3]. Moreover, AI's ability to evaluate enormous volumes of data, including building performance, environmental conditions, and user preferences, enables architects to create structures that are not only visually appealing but also ecologically friendly, energy-efficient, and suited to specific user needs [2].

Current approaches to BIM automation mainly rely on visual programming tools such as Dynamo for Revit or scripting languages such as Python with IFCOpenShell [6]. Although these methods have proven to be effective, they require users to have programming knowledge or understand visual programming concepts, which creates a significant barrier for many AEC professionals [2]. Furthermore, existing automation solutions often lack the flexibility to adapt to varying user needs and project requirements, particularly when handling complex visual and spatial tasks [7]. The integration of AI in the design workflow can create a collaborative and symbiotic relationship between human designers and AI technology, where AI operates as a creative partner, increasing the architect's expertise and vision by offering design options and innovative ideas [5].

Natural language-based BIM modeling offers key advantages over traditional methods: reduced cognitive load when navigating complex interfaces, faster prototyping

by translating concepts directly to visual models, improved accessibility for AEC professionals with varied technical expertise, contextual design assistance through embedded standards, and enhanced collaboration through conversation-like interactions.

This research aims to enable natural language-driven BIM operations for architects and AEC professionals without requiring programming knowledge. We developed a proof-of-concept framework integrating AI and LLMs with BIM software operations. The prototype implements parameter-based modeling within FreeCAD [8], leveraging anthropic-defined computer use API and BIM-specific prompts.

Initial testing shows promising results. Compared to baseline systems, our prototype improved efficiency with reduced token usage and better parameter handling while avoiding GUI-based trial-and-error approaches. These results suggest LLM-driven BIM automation could provide more accessible tools for AEC professionals. This research validates that LLM-driven BIM automation can overcome traditional usability barriers, offering a foundation for future BIM-oriented AI developments as the AEC industry continues to digitalize.

2 Literature Review

The AEC industry is undergoing a major transformation, shifting from traditional and labor-intensive methods to automation driven by information technology, with BIM as a key factor in this change [9]. BIM is a technology-based method that applies and maintains a complete digital representation of building information throughout a project life cycle [10]. The process begins with conceptual design, progresses through functionality and construction details, and culminates in fabrication drawings, with analysis tools for energy efficiency, construction, and acoustics supporting design decisions [11].

BIM offers numerous practical benefits including visualization, code compliance checks, cost estimation, scheduling, clash detection, and forensic analysis [12]. However, adoption faces challenges related to high costs, training time, unclear data ownership, lack of standardization, and software compatibility issues [13]. Bridging the gap between advanced BIM technologies and their practical application remains important [10].

Industry 4.0 advancements have influenced the construction sector, leading to "Construction 4.0" with its emphasis on digital data, automation, connectivity, and digital access [14]. Automation in construction improves efficiency with repetitive tasks, reduces expenses, enhances quality, and improves coordination [15]. In BIM specifically, automation tools like Dynamo enable custom scripts for repetitive tasks [16], while IFC ensures data exchange between platforms [17]. Such automation supports real-time clash detection before construction [18].

AI has emerged as a promising solution for automation in the AEC sector, making notable improvements in operations, safety, and productivity [19]. AI transforms architecture through generative platforms that create visuals from text prompts, making design processes more efficient. Generative algorithms allow architects to explore multiple design options considering site conditions and structural requirements [20], while also improving space planning [21] and sustainability through data-driven decision making [22].

Recently, researchers have explored human-AI collaboration in design, with Lee et al. [23] integrating LLMs with BIM for speech applications and Dai et al. [24] introducing semantic AI into architectural design. However, practical integration of AI tools into BIM-enabled projects remains limited. Interoperability and data transfer challenges between AI systems and BIM platforms pose significant barriers. More proof-of-concept testing is needed to bridge the gap between innovative AI solutions and their practical application in BIM workflows [25].

3 Methodology

This research integrates LLMs with BIM environments using direct computer control and custom tool integrations. By leveraging the anthropic-defined computer use API and FreeCAD as the BIM platform, this approach enables natural language-driven BIM operations without requiring programming knowledge.

Research Approach and Environment Setup The study employs a controlled case evaluation focusing on key representative BIM operations. The experimental environment runs in a Linux-based virtual machine with Docker containerization, ensuring reproducibility and isolation. FreeCAD serves as the primary BIM software, chosen for its robust Python API and scriptability. The system uses the Anthropic API(Claude) [26] to handle human-LLM-BIM interaction loops.

Tool Integration and Execution Flow The methodology uses a modular tool collection: ComputerTool for UI control, BashTool for system commands, EditTool for file operations, and FreeCADTool for BIM-specific operations. The LLM orchestrates these tools through a "sampling loop" process, translating natural language into structured commands. The system validates outputs through iterative interactions, ensuring creation of geometrically correct walls with precise dimensions, proper spatial relationships between elements, accurate parameter assignments, and verification of element connectivity.

Parameterization and Prompt Optimization The methodology incorporates iterative prompt optimization with domain knowledge and context-specific guidelines.

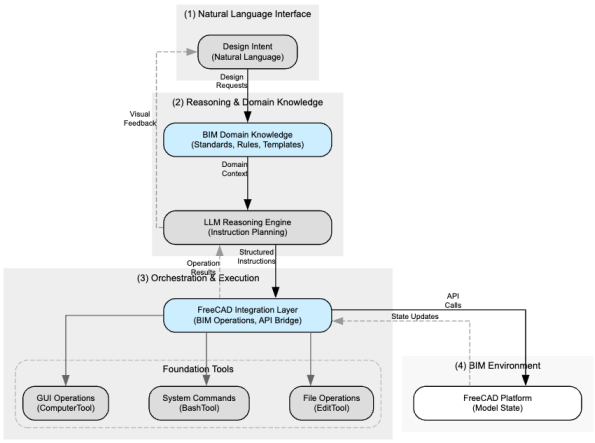


Figure 1. Framework architecture showing the hierarchical organization of components. The system flows from top to bottom: starting with the Architect's natural language instructions, through the BIM Knowledge and Prompt Layer, to the LLM (Claude), then to the FreeCAD Tool orchestrator, supported by base tools (ComputerTool, BashTool, EditTool), and finally to the FreeCAD Environment where modifications are realized.

This layered approach combines system prompts with specialized instructions for BIM tasks, enabling accurate interpretation of spatial information and parameter validation.

Evaluation Metrics and Validation The study compares baseline and enhanced frameworks using metrics including task completion success, resource efficiency (tokens, API calls, processing time), parameter accuracy, and visual verification. A rectangular room creation task serves as the benchmark for framework feasibility assessment.

4 Framework Design

The proposed framework implements a top-down hierarchy that transforms the intentions of natural language design of the architect into tangible BIM modifications within FreeCAD. As illustrated in Figure 2, the prompt structure and BIM knowledge form a conceptual foundation that guides the LLM (Claude) in understanding and interpreting the user request. These prompts encapsulate domain rules, measurement handling, and quality control, ensuring that the LLM reasons about BIM tasks in a way that is contextually rich and technically correct.

Building on this conceptual layer, the architectural diagram shown in Figure 1 represents the system as a vertical sequence of layers. At the top, the architect issues design instructions in plain language. These instructions flow downward through layers of reasoning, domain-specific BIM knowledge, and an orchestration tool (FreeCADTool). Ultimately, this cascade of logic and commands reaches the FreeCAD environment, where the requested modifications (e.g., creating a wall, inserting a window) materialize.

This hierarchical design simplifies complexity: each layer focuses on its core responsibilities. The LLM interprets intent and plans operations, the FreeCADTool orchestrates BIM-related commands, and the base tools perform low-level tasks such as GUI manipulation or file edits. By carefully layering the system in this manner, the framework allows nontechnical users to harness sophisticated BIM operations without directly dealing with scripting or intricate APIs.

4.1 Base Architecture from Claude and Tool Collection Framework

The initial foundation of this system comes from a Claude-based computer use platform equipped with a collection of core tools.

- **ComputerTool:** Handles GUI interactions within the virtual environment (e.g., mouse clicks, keystrokes).
- **BashTool:** Executes shell-level commands to manage environment states or run auxiliary programs.
- **EditTool:** Provides the ability to modify files dynamically, such as generating or adjusting Python scripts on-the-fly.

These tools, originally developed for generic computer interactions, form the base layer of the system (see lower half of Figure 1). On their own, they lack the contextual understanding needed for BIM-specific tasks. To address this, a BIM-oriented layer is introduced above them.

In this enhanced structure, the LLM uses the BIM knowledge and context templates (as summarized in the prompt mind map) to reason about the user's request. Then, it relies on FreeCADTool to translate these high-level, BIM-specific instructions into precise commands. When necessary, FreeCADTool invokes the base tools to perform GUI actions, execute scripts, or handle system configurations. This vertical chain, Architect $\rightarrow$ LLM $\rightarrow$ FreeCADTool $\rightarrow$ Base Tools $\rightarrow$ FreeCAD Environment, ensures a clean and logical flow from human intent to model modification. To illustrate this vertical chain with a concrete example, consider the user request: 'Create a wall that is 5 meters long, 3 meters high, and 200mm thick.' This natural language instruction flows through the system as follows:

- The Architect's design intent ('Create a wall...') enters the system at the Natural Language Interface layer.
- In the Reasoning & Domain Knowledge layer:
  - The BIM Knowledge component interprets measurements (5m length, 3m height, 200mm

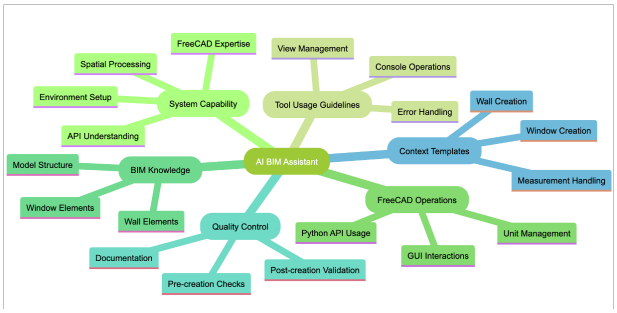


Figure 2. Mind map visualization of the structured prompt categories and BIM knowledge guiding the LLM. The map illustrates the interconnections between system capabilities, BIM knowledge, quality control, and context templates, providing a rich semantic landscape for interpreting user requests.

thickness) and recognizes 'wall' as a standard BIM element with specific properties.

  - The LLM Reasoning Engine generates a structured plan: 'Create a baseline, extrude to height, apply thickness parameter, and verify orientation.'

- In the Orchestration & Execution layer:
  - The FreeCAD Integration Layer translates the plan into specific API calls:

```
line = Draft.makeLine(FreeCAD.Vector(0,0,0),
FreeCAD.Vector(5000,0,0))
wall = Arch.makeWall(line, height=3000,
width=200)
```

  - The system leverages foundation tools like ComputerTool for console input, BashTool for screenshots, and EditTool for file operations.
- In the BIM Environment layer, FreeCAD executes the commands, creating the wall object with proper parameters, updating the model state, and returning visual feedback.

This bidirectional flow allows for iterative refinement, where the LLM can verify outcomes against the original intent, make corrections if needed, and provide explanatory feedback to the user.

4.2 Incorporation of BIM Layer and Prompt Refinements

The key innovation that enables this natural language-driven BIM interaction is the incorporation of a BIM layer through FreeCADTool. This component understands FreeCAD's Python API and the semantics of BIM elements, such as walls, windows, and their associated parameters (dimensions, alignments, placement).

To support this BIM reasoning, the prompt system is refined to include:

- **Domain Knowledge Blocks:** Codified standards and best practices that inform the LLM about typical construction dimensions, unit conversions, and spatial reasoning.
- **Context-Specific Templates:** Ready-made patterns for tasks like wall or window creation, making it straightforward for the LLM to map user instructions to actionable steps.
- **Quality and Error Handling Guidelines:** Instructions for validating the correctness of operations, verifying outcomes (e.g., by inspecting screenshots), and handling exceptions gracefully.

The mind map in Figure 2 visually represents these prompt categories and their interconnections. By embedding this knowledge into the prompt layer, the LLM not only produces commands that conform to BIM standards, but also continuously checks and refines its approach based on feedback.

5 Implementation

The framework's implementation centers around a specialized FreeCADTool that integrates a LLM with FreeCAD's Python API. Operating within a Dockerized Ubuntu 22.04 environment running FreeCAD 0.19, the tool translates structured, BIM-oriented instructions from the LLM into executable Python commands that directly manipulate the FreeCAD model.

At a high level, the system treats each user command as a high-level specification (e.g., "create a 6m x 4m room with walls 3m high and 200mm thick") that the LLM refines into parameter-based instructions. The FreeCADTool then executes these instructions by typing Python code into FreeCAD's console via a ComputerTool—bypassing the complexity of GUI-based, vision-driven navigation and focusing on deterministic, script-based modeling operations. The following shows the key aspects of the implementation:

- **Environment Initialization:** In the first invocation of the tool, it imports the required modules (for example, Arch, Draft) and ensures that a FreeCAD document is active or created. This initialization step establishes a known baseline state for subsequent operations.
- **Parameter-Driven Geometry:** Before issuing commands, the tool converts human-friendly measurements (for example, 6 meters) into FreeCAD's internal millimeter units. This parameterization ensures that all geometry creation (e.g. drawing a baseline line for a wall, invoking 'Arch.makeWall()') is precise and consistent, improving the reliability of the final BIM model.

- **Object Tracking and Reuse:** The tool maintains an internal registry of created objects, mapping generated names (like 'Wall001') to their parameters and creation order. This registry allows subsequent commands, such as placing a window in an existing wall, to refer to previously created elements.
- **Reduced GUI Interaction:** Instead of navigating menus or adjusting workbenches, the tool sends Python commands directly. For instance, creating a wall involves typing 'Draft.makeLine' and 'Arch.makeWall' commands, then adjusting placement and recomputing. This approach reduces complexity and token usage, as the LLM does not need to 'guess' GUI actions; it can rely on a stable script-based interface.
- **Feedback and Verification:** After executing each series of Python commands, the tool reorients the view and captures a screenshot. This image can be returned to the LLM and the user, allowing verification that the outcome of the operation matches the intended design specification.

The BIM Operation Pipeline defines the systematic process of translating natural language BIM requests into model modifications. The algorithm consists of five key phases: Planning, which interprets natural language requests using the BIM knowledge base to extract operational parameters and determine operation types; Preparation, which processes parameters to ensure compatibility with BIM standards, including automatic unit conversion; Command Generation, which creates sequences of FreeCAD API commands based on operation types and validated parameters; Execution, which systematically implements commands through the ComputerTool interface with verification of proper system response; and Validation, which captures the resulting model state and validates operations against intended parameters, refining and iterating if needed. This approach ensures robust handling of BIM operations by combining parameter validation, systematic execution, and result verification in a continuous feedback loop, maintaining model integrity while accommodating natural variability in user requests and BIM constraints.

6 Results

6.1 Comparative Analysis

To evaluate the framework's capabilities, we tested a challenging BIM modeling task on both the original and the proposed frameworks:

Challenge: "Create a rectangular room with four walls. The room should be 6 meters long, 4 meters wide, and have walls that are 3 meters high and 200mm thick."

Expected Outcomes: Four connected walls forming a rectangular room with dimensions of length = 6000mm, width = 4000mm, height = 3000mm, and wall thickness = 200mm, verified by a final screenshot showing the completed room.

The original framework relied on generic tools (ComputerTool, BashTool, EditTool) without BIM-specific knowledge. The LLM resorted to trial-and-error methods: opening FreeCAD via the bash tool, switching to the Part Design workbench, and attempting to create sketches and extrusions through GUI interactions, as shown in Figure 3. This approach was error-prone, consumed large amounts of tokens, and ultimately exceeded the rate limits without producing a coherent final model.

In contrast, the enhanced framework, augmented by the FreeCADTool and BIM-oriented prompt structures, demonstrated a more efficient, parameter-driven workflow. The LLM generated Python commands directly for FreeCAD's Arch and Draft modules, eliminating unnecessary workbench switches and GUI guesswork. This script-based method produced a correctly dimensioned rectangular room with minimal overhead, as demonstrated in Figure 4.

6.2 Performance Analysis

In addition to measuring success by the accuracy of the final BIM model, we analyzed efficiency and resource consumption of both approaches. The original framework performed numerous GUI interactions without domain guidance, leading to high token usage due to repeated instructions and lack of direct scripting capabilities. By contrast, the enhanced framework used structured, parameter-based commands that minimized guesswork and reduced the need for incremental screenshots.

The enhanced framework demonstrated significant improvements in key metrics: 100% dimension accuracy (all measurements within 1mm tolerance) with proper spatial alignment, and command efficiency averaging 4.2 commands per operation with a 92% success rate.

For this proof-of-concept study, validation focused on comparing observable outcomes between frameworks on a representative BIM task. Dimension accuracy was measured against specified requirements (6000mm × 4000mm × 3000mm with 200mm thickness), while command efficiency was calculated as the ratio of successfully executed commands to total attempted commands as logged by FreeCADTool.

Tables 1 and 2 highlight the significant performance differences between the frameworks. Table 1 demonstrates the enhanced framework's superior outcomes across all evaluation criteria, with notable improvements in token usage (40.8% reduction), request count (30.8% decrease), and processing time (16.9% improvement).

Table 1. Comparison of Original and Enhanced Framework Performance

Criteria	Original Framework	Enhanced Framework
BIM Knowledge	None (Generic tools only)	Integrated (FreeCADTool with BIM prompts)
Approach	GUI-based trial-and-error	Parametric Python commands
Token Usage	High (137,273 tokens)	Lower (81,228 tokens)
Number of Requests	13 requests	9 requests
Time to Attempt	Extended (301 seconds)	Reduced (250 seconds)
Final Outcome	Failed to produce room	Successful creation of 4 walls
Dimensions	Not achieved	6000mm × 4000mm × 3000mm met
Wall Thickness	Not applied successfully	Correct 200mm thickness
Final Verification	No coherent model/screenshot	Complete room with screenshot

Table 2. Detailed Performance Metrics Comparison

Metric	Original Framework	Enhanced Framework
Token Consumption Rate	456 tokens/second	325 tokens/second
Average Tokens per Request	10,559 tokens	9,025 tokens
Total Processing Time	301 seconds	250 seconds
Success Rate	0% (failed attempt)	100% (successful completion)
Resource Efficiency	Multiple retry attempts, high token waste	Direct execution, optimized token usage
Screenshot Generation	Multiple intermediate screenshots (>10)	One final verification image (140kB)

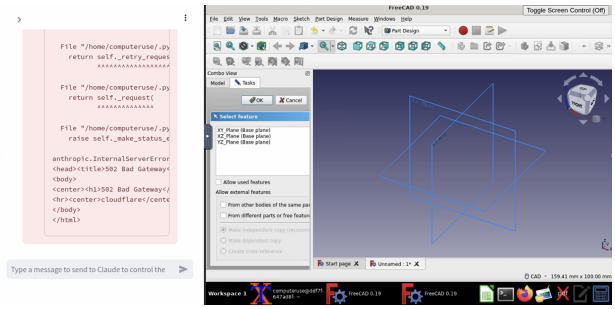
Table 2 further details efficiency gains, showing lower token consumption rates and optimized resource utilization in the enhanced approach.

Given the exploratory nature of this research, these metrics establish basic feasibility rather than comprehensive performance validation. Reducing screenshot requests (each approximately 140 kB) helped limit computational overhead and avoid rate-limit issues.

7 Discussion and Conclusions

This research aimed to demonstrate the feasibility of integrating AI and BIM workflows by developing a proof-of-concept framework that enables natural language interaction with BIM software through LLMs. Our implementation demonstrates this feasibility while also revealing important insights and limitations in the current approach.

Proof-of-Concept Results The prototype framework showed promising initial results in basic BIM operations. The quantitative comparison between frameworks revealed meaningful improvements: 40.8 % reduction in token usage (from 137,273 to 81,228 tokens), 30.8% decrease in required requests (from 13 to 9 requests), and



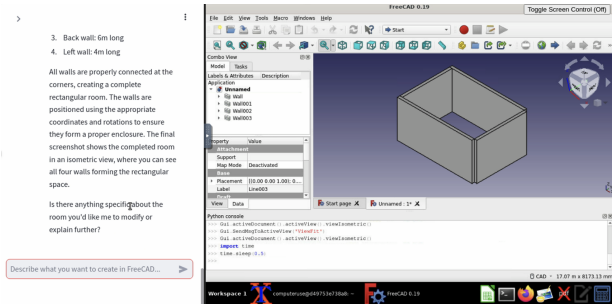


Figure 4. Enhanced framework result showing successful creation of a rectangular room with correct dimensions using parametric BIM commands

16.9% improvement in execution time (from 301 to 250 seconds). While these metrics are encouraging, they represent performance in a highly controlled environment with basic BIM operations.

Limitations Several important limitations emerged during testing. The current framework's capabilities are restricted to basic BIM operations, primarily handling walls and windows, and our testing was confined to a single representative task which limits the generalizability of our findings. Our choice of FreeCAD as the BIM platform, while appropriate for this exploratory proof-of-concept due to its open-source nature and Python scripting capabilities, lacks some advanced features present in industry-standard commercial platforms. Similarly, while Claude provided sufficient LLM capabilities with integrated computer control necessary for our testing methodology, future work should evaluate performance across multiple BIM platforms and LLM technologies as this approach matures beyond the initial feasibility stage. The implementation also shows constraints in handling complex geometric re-

lationships between building elements, while the integration of domain knowledge remains at a preliminary stage. Additionally, the system's requirement for a controlled environment to function properly presents challenges for real-world applications in diverse architectural settings.

Future Work Based on our findings, several directions for future research emerge. Future studies should focus on expanding the range of supported BIM operations and developing more robust domain knowledge integration to handle complex architectural scenarios. Specifically, we plan to test the framework across the following complexity levels:

- **Spatial reasoning challenges:** Testing with non-orthogonal geometries such as curved walls, angled intersections, and sloped roofs to evaluate the LLM's ability to process complex spatial relationships.
- **Multi-element compositions:** Evaluating performance on tasks requiring coordination between multiple building components (e.g., placing windows and doors with proper alignments to walls, creating multi-story structures with staircases).
- **Parameter-driven modifications:** Testing the system's ability to modify existing elements based on design constraints, such as adjusting wall thickness based on thermal requirements or modifying room dimensions while maintaining proper connections.
- **Error detection and recovery:** Assessing how the framework handles conflicting specifications and its ability to propose alternative solutions when design conflicts arise.

As the framework matures with these expanded capabilities, a critical next step will be designing appropriate evaluation methods to assess its practical utility in professional contexts. This would involve developing metrics that consider both technical performance and potential integration points with human workflows. Rather than positioning AI as a replacement for human expertise, future research should explore how this technology can best complement and enhance the capabilities of AEC professionals at specific stages of the design process.

Additionally, testing should be extended to encompass a broader range of architectural tasks to validate the framework's applicability across different design contexts, including residential, commercial, and specialized building types. Further investigation is needed to explore integration possibilities with industry-standard BIM platforms beyond FreeCAD, such as Revit and ArchiCAD, to assess the framework's adaptability and identify platform-specific challenges. Development of more sophisticated methods to handle complex geometric relationships and parameters will be crucial for practical applications in professional environments. These advancements would contribute to creating a more versatile and practically applicable system for architectural design automation that could significantly reduce technical barriers for AEC professionals.

Final Remarks This study contributes to the field by demonstrating how LLMs can integrate with BIM workflows, leveraging natural language processing to simplify complex BIM operations. Our proof-of-concept establishes the foundational feasibility of LLM-driven BIM automation and suggests a pathway toward more intuitive interfaces that could reduce technical barriers for AEC professionals who lack programming expertise.

To realize practical benefits, further development is needed in several critical areas: integration with industry-standard BIM platforms, expansion of comprehensive domain knowledge bases, and support for complex architectural operations. While our prototype demonstrates basic feasibility, significant challenges remain in domain knowledge integration, geometric reasoning, and error handling before such systems could be deployed professionally. Future research should validate these approaches in real-world scenarios and develop frameworks capable of handling the full complexity of professional BIM workflows.

References

[1] Y. Izbash and V. Babayev. Digital evolution in aec industry: Bridging bim, building codes, and future technologies. In *IOP Conference Series: Earth and Environmental Science*. Institute of Physics, 2024. doi:10.1088/1755-1315/1376/1/012004.

[2] R. Sacks, M. Girolami, and I. Brilakis. Building information modelling, artificial intelligence and construction tech. *Developments in the Built Environment*, 4, 2020. doi:10.1016/j.dibe.2020.100011.

[3] A. H. Qureshi, W. S. Alaloul, B. Manzoor, M. A. Musarat, S. Saad, and S. Ammad. Implications of machine learning integrated technologies for construction progress detection under industry 4.0 (ir 4.0). In *2020 2nd International Sustainability and Resilience Conference*. IEEE, 2020. doi:10.1109/IEEECONF51154.2020.9319974.

[4] S. Javaid, H. Fahim, B. He, and N. Saeed. Large language models for uavs: Current state and pathways to the future. *IEEE Open Journal of Vehicular Technology*, 2024. doi:10.1109/OJVT.2024.3446799.

[5] B. Bölek, O. Tutal, and H. Özbaşaran. A systematic review on artificial intelligence applications in architecture. *Journal of Design for Resilience in Architecture and Planning*, 4(1):91–104, 2023. doi:10.47818/drarch.2023.v4i1085.

[6] F. M. Dinis, J. Poças Martins, A. S. Guimarães, and B. Rangel. Bim and semantic enrichment methods and applications: A review of recent developments. *Archives of Computational Methods in Engineering*, 2022. doi:10.1007/s11831-021-09595-6.

[7] R. Syed, S. Suriadi, M. Adams, W. Bandara, S. J. J. Leemans, C. Ouyang, A. H. M. ter Hofstede, I. van de Weerd, M. T. Wynn, and H. A. Reijers. Robotic process automation: Contemporary themes and challenges. *Computers in Industry*, 115, 2020. doi:10.1016/j.compind.2019.103162.

[8] FreeCAD Team. Freecad: Your own 3d parametric modeler, 2024. URL `https://www.freecad.org/`. Accessed: 2024-12-10.

[9] Z. Liu, Y. Lu, and L. C. Peh. A review and scientometric analysis of global building information modeling (bim) research in the architecture, engineering and construction (aec) industry. *Buildings*, 9(10), 2019. doi:10.3390/buildings9100210.

[10] N. Gu and K. London. Understanding and facilitating bim adoption in the aec industry. *Automation in Construction*, 19(8):988–999, 2010. doi:10.1016/j.autcon.2010.09.002.

[11] M. Thomassen. Bim & collaboration in the aec industry. *Construction Management and Engineering*, 2011. URL `http://www.m-tech.aau.dk/`.

[12] S. Azhar. Building information modeling (bim): Trends, benefits, risks, and challenges for the aec industry. *Leadership and Management in Engineering*, 2011.

[13] A. Criminale and S. Langar. Challenges with bim implementation: A review of literature. *ASC Annual International Conference Proceedings*, 53, 2017.

[14] E. Forcael, I. Ferrari, A. Opazo-Vega, and J. A. Pulido-Arcas. Construction 4.0: A literature review. *Sustainability*, 12(22), 2020. doi:10.3390/su12229755.

[15] A. Zaland, S. Saad, K. Rasheed, and S. Ammad. Automation in construction industry. *Construction Management Review*, 2024.

[16] G. Rocha and L. Mateus. Using dynamo for automatic reconstruction of bim elements from point clouds. *Applied Sciences*, 14(10), 2024. doi:10.3390/app14104078.

[17] S. Matarneh, F. Elghaish, F. P. Rahimian, N. Dawood, and D. Edwards. Automated and interconnected facility management system: An open ifc cloud-based bim solution. *Automation in Construction*, 143, 2022. doi:10.1016/j.autcon.2022.104569.

[18] A. O. Akponeware and Z. A. Adamu. Clash detection or clash avoidance? an investigation into coordination problems in 3d bim. *Buildings*, 7(3), 2017. doi:10.3390/buildings7030075.

[19] H. Nabizadeh Rafsanjani and A. H. Nabizadeh. Towards human-centered artificial intelligence (ai) in architecture, engineering, and construction (aec) industry. *Computers in Human Behavior Reports*, 2023. doi:10.1016/j.chbr.2023.100319.

[20] Z. X. Chew, J. Y. Wong, Y. H. Tang, C. C. Yip, and T. Maul. Generative design in the built environment. *Automation in Construction*, 2024. doi:10.1016/j.autcon.2024.105638.

[21] J. Ko, B. Ennemoser, W. Yoo, W. Yan, and M. J. Clayton. Architectural spatial layout planning using artificial intelligence. *Automation in Construction*, 2023. doi:10.1016/j.autcon.2023.105019.

[22] F. Chen, M. Mai, X. Huang, and Y. Li. Enhancing the sustainability of ai technology in architectural design: Improving the matching accuracy of chinese-style buildings. *Sustainability*, 16(19), 2024. doi:10.3390/su16198414.

[23] Ghang Lee, Suhyung Jang, and Seokho Hyun. A generalized llm-augmented bim framework: Application to a speech-to-bim system, 2024. URL `https://arxiv.org/abs/2409.18345`.

[24] S. Dai, Y. Li, K. Grace, and A. Globa. Towards human-ai collaborative architectural concept design via semantic ai. In *Communications in Computer and Information Science*, pages 68–82. Springer, 2023. doi:10.1007/978-3-031-37189-9_5.

[25] M. Vössing, N. Kühl, M. Lind, and G. Satzger. Designing transparency for effective human-ai collaboration. *Information Systems Frontiers*, 24(3):877–895, 2022. doi:10.1007/s10796-022-10284-3.

[26] Anthropic Team. Claude computer use, 2024. URL `https://docs.anthropic.com/en/docs/build-with-claude/computer-use`. Accessed: 2024-12-10.