2026 European Conference on Computing in Construction
Corfu, Greece
July 12–15, 2026

A Self-Verification Framework Toward Reliable Text-to-BIM Generation

Tobias Sesterhenn[1], Bharathi Kannan Nithyanantham[2], Stefan Lüdtke[2], and Christian Bartelt[1]
[1]Institute for Software and Systems Engineering, TU Clausthal, Germany
[2]Institute for Visual and Analytic Computing, University of Rostock, Germany

Abstract

Recent advances in generative AI have enabled Text-to-BIM systems that generate building models from natural-language prompts; however, these systems typically lack automated verification, requiring users to manually inspect and iteratively correct errors. To increase the reliability of Large Language Model (LLM)-generated BIM outputs, we propose a Text-to-BIM workflow with an integrated self-verification component. For each prompt, the system derives two complementary validations: an Information Delivery Specification (IDS) for rule-based checking, and an LLM-driven code-based verification for requirements not expressible in IDS. The resulting feedback is used to iteratively refine the IFC model in a closed loop. The framework is implemented and evaluated in a case study, demonstrating that the self-correcting process improves model quality.

Introduction

Text-to-BIM systems generate Building Information Models (BIM) from natural language instructions (Dong et al., 2025; Deng et al., 2025), but despite their capabilities, current approaches lack a task-specific self-verification component that ensures the final BIM model actually meets the request of the human user. For example, Text2BIM (Du et al., 2026) includes a rule-based model checker, but all checks are predefined and may lose task-specific requirements. Consequently, users must manually inspect IFC models and iteratively correct deviations, limiting reliability and scalability of such systems.

To address this, inspired by self-verification (Gou et al., 2024), we propose a closed-loop, LLM-assisted Specification-Modification-Verification framework. For each prompt, a *Specifier* agent derives two complementary specifications: (i) an automatically checkable Information Delivery Specification (IDS) encoding all formally verifiable, schema-level constraints, and (ii) a higher-level specification capturing semantic and contextual requirements that are evaluated by a *Verifier* agent. This allows a *Modifier* agent creating a BIM model to iteratively receive automated feedback by the *Verifier*, to finally output a model that meets the given specification. Our contributions are as follows:

- A fully automated Text-to-BIM framework with integrated self-verification.
- A novel Text-to-IDS module that turns natural-language requirements into valid IDS constraints.
- A simple case study demonstrating the framework.

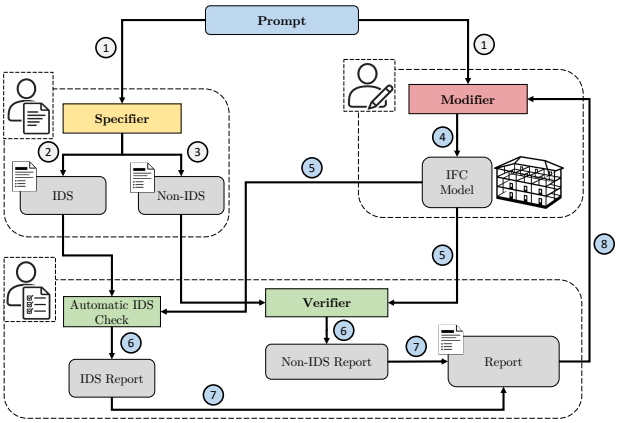

Figure 1: The Modifier-Specifier-Verifier Framework. Steps 4-8 are colored blue to indicate they are repeated in each iteration of the Modifier-Verifier Loop. The Specifier is only required in the first iteration to produce the specification documents.

Framework

As shown in Figure 1, the framework coordinates three agents – the *Specifier*, *Modifier*, and *Verifier* – within a closed-loop architecture to translate a user's design prompt into an iteratively refined IFC model. Given a human prompt (①), the *Specifier* Agent derives two complementary artifacts: (i) an IDS document encoding all formally expressible, schema-level requirements (e.g., properties, simple relations) (②), and (ii) a less formal non-IDS specification capturing constraints that cannot be represented in IDS form, such as geometric, topological, or complex relational conditions (③).

The *Modifier* Agent generates and iteratively updates an IFC model based on the prompt and – in following iterations – subsequent feedback (④). In each iteration, validation on the current IFC model is then performed at two levels (⑤). An automated IDS checker evaluates the model against the IDS document, producing an IDS Report (⑥). In parallel, the *Verifier* Agent operationalizes the non-IDS specification by generating and executing custom verification code (i.e., programmatic checks over the IFC structure), also producing a report. Both reports are consolidated into a unified feedback report (⑦), which is returned to the Modifier Agent to guide model revision (⑧). The loop continues until the IFC Model satisfies both the formal IDS constraints and the non-IDS requirements, or until a predefined iteration limit is reached.

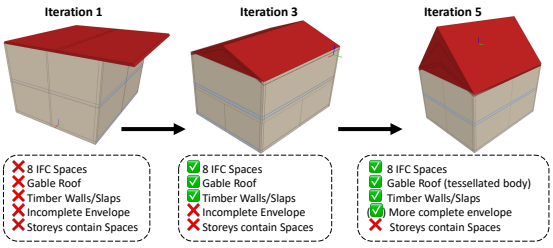

Figure 2: Iterative refinement in the case study. The boxes indicate compliance with individual IDS and non-IDS requirements at each iteration of the self-verification loop.

Case Study

To demonstrate the framework, we conduct a case study using the prompt: *"Create a simple but realistic house, 4 rooms per level, 2 levels, made out of wood. The roof should be a gable roof."*[1]

We use GPT-5.2 for five iterative refinement cycles. The *Modifier* and *Verifier* agents are equipped with a toolbox based on MCP4IFC (Nithyanantham et al., 2025), enabling iterative programmatic IFC model construction and querying. Concretely, both agents can interact with a python interpreter and run code using either available functions from the toolbox (e.g. `create_wall()`) or by writing own `ifcopenshell-python`[2] code. In each iteration the agent is allowed to make multiple tool calls and to incorporate the results into its reasoning process, as formalized by the ReAct framework (Yao et al., 2022). Automated IDS validation is performed using the open-source `ifctester`[2] implementation.

As illustrated in Figure 2, the system progressively resolves detected violations. For instance, the IDS requirement of eight `IFCSpace` entities fails in Iteration 1 and is satisfied by Iteration 3. Similarly, the geometric verification of the gable roof initially fails and is corrected by Iteration 3, together with the assignment of the material attribute "*Timber*" to relevant wall and slab elements. By Iteration 5, the roof geometry and overall building envelope are further refined.

Discussion

The case study provides a proof of concept for the proposed framework; however, a comprehensive evaluation is required to assess its robustness and generalizability. In the presented experiment, several limitations became apparent. For instance, the system exhibited difficulties in reliably deleting existing elements and failed to resolve certain constraint violations across iterations, such as the missing relationship between the eight `IFCSpace` entities and their corresponding `IFCBuildingStorey` elements. Moreover, complete coverage of all formal and non-IDS requirements during specification generation cannot be guaranteed, and any errors or omissions at this stage directly impact subsequent validation and refinement.

[1] Our code is publicly available at github.com/Tsesterh/Text2BIM-Self-Verification.

[2] `https://docs.ifcopenshell.org`

Conclusion & Future Work

We presented an agent-based Text-to-BIM framework centered on self-verification. For each prompt, the system derives checkable constraints and leverages automated feedback to iteratively revise the IFC model until detected violations are minimized or resolved. The simple case study suggests that this specification-modification-verification loop improves model quality and reliability, while also highlighting the need for further evaluation and refinement to ensure robustness and broader applicability. Future work should therefore focus on systematic evaluation and strengthening of the individual agents. The *Specifier* agent could be assessed through a dedicated Text-to-IDS benchmark to measure completeness and correctness of constraint extraction. The *Verifier* agent may benefit from integrating deterministic geometry and topology checks to reduce reliance on purely reasoning-based validation. In addition, hybrid validation strategies should be explored, combining LLM-generated specifications with predefined or human-authored rule sets to improve reliability.

Acknowledgements

This research was funded by the German Federal Ministry of Research, Technology and Space (BMFTR) via the Show2Instruct project (Grant No. 01MK25008D).

References

Deng, Z., Du, C., Nousias, S., and Borrmann, A. (2025). Bimgent: Towards autonomous building modeling via computer-use agents. arXiv preprint arXiv:2506.07217.

Dong, Y., Zhan, Z., Hu, Y., Doe, D. M., and Han, Z. (2025). Ai bim coordinator for non-expert interaction in building design using llm-driven multi-agent systems. Automation in Construction, 180:106563.

Du, C., Esser, S., Nousias, S., and Borrmann, A. (2026). Text2bim: Generating building models using a large language model-based multiagent framework. Journal of Computing in Civil Engineering, 40(2):04025142.

Gou, Z., Shao, Z., Gong, Y., yelong shen, Yang, Y., Duan, N., and Chen, W. (2024). CRITIC: Large language models can self-correct with tool-interactive critiquing. In The Twelfth International Conference on Learning Representations.

Nithyanantham, B. K., Sesterhenn, T., Nedungadi, A., Peral Garijo, S., Zenkner, J., Bartelt, C., and Lüdtke, S. (2025). MCP4IFC: IFC-based building design using large language models. arXiv preprint arXiv:2511.05533.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. (2022). React: Synergizing reasoning and acting in language models. In The eleventh international conference on learning representations.