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

Teaching BIM as a Computational, Queryable, and Analyzable Model: A Project-Based Framework Integrating APIs, Programming, and Early Performance Reasoning

Tobias Maile, Burcu Güldür Erkal
Technical University of Applied Sciences Augsburg, Germany

Abstract

Building Information Modeling (BIM) education in architecture, engineering, and construction (AEC) remains largely tool-centric, limiting students' understanding of data quality, automation, and performance reasoning. This paper presents a project-based educational framework that treats BIM models as computational, data-driven artifacts. Through programming, API-based model access, simplified performance analyses, and human-in-the-loop large language model (LLM) support, students develop computational BIM literacy through learning-by-doing. Student feedback indicates improved awareness of modeling consequences, data quality, and early performance assessment. The framework offers a transferable approach for computationally oriented AEC curricula and reframes BIM education as data quality education.

Introduction

Building Information Modeling (BIM) has become a central methodology in architecture, engineering, and construction (AEC), serving as a shared digital representation for design, analysis, construction, and operation. While BIM is gaining momentum in professional practice, its integration into higher education remains uneven. Many curricula still emphasize software operation and predefined workflows rather than conceptual understanding of BIM models as structured, queryable, data-driven representations (Succar, 2009; Eastman et al., 2018).

A recurring challenge in BIM education is the limited awareness of **data quality and data consistency**. Students frequently interact with BIM models as geometric artifacts, without fully understanding how model structure, parameter completeness, or semantic consistency influence downstream use. As a result, modeling errors or implicit assumptions often remain unnoticed until they manifest as problems in later project stages, such as performance analysis, quantity take-off, or interdisciplinary coordination. This delayed feedback makes it difficult for students to develop a causal understanding of how early modeling decisions propagate through digital workflows.

Automation and computational workflows represent another critical gap. Scripting and model-based automation are often introduced as advanced or optional topics, reinforcing the perception of computational methods as expert-only skills. This contributes to black-box tool usage and reliance on default software settings, limiting transparency and reproducibility of model-based reasoning (Borrmann et al., 2018).

Furthermore, BIM education is commonly dominated by **lecture-driven teaching formats**. While lectures convey conceptual knowledge, they are often insufficient for developing experiential understanding of how digital models behave under computational use. Multidisciplinary analyses, such as energy and structural assessment, are typically taught as isolated workflows, obscuring their shared reliance on a common underlying model.

Multidisciplinary performance analysis is frequently introduced as a set of isolated tools or domain-specific workflows. Structural, energy, and environmental analyses are often treated independently, each relying on separate model abstractions and software environments. This fragmentation obscures the fact that all analyses ultimately depend on the same underlying building model, albeit represented through different abstractions such as analytical models for structural reasoning or simplified thermal representations for energy assessment.

Summarizing the current challenges in education:

- Limited awareness of data quality and data consistency
- Insufficient understanding of how modeling decisions propagate errors
- Automation perceived as expert-only or tool-specific
- Predominance of front-loaded, lecture-based teaching
- Fragmentation of multidisciplinary performance analysis
- Lack of model transparency and explainability
- Over-reliance on software defaults and black-box workflows

Together, these challenges point to a fundamental educational gap: students are rarely taught to understand BIM models as **computational, data-driven artifacts** that can support automation, reasoning, and early performance exploration across multiple domains. Addressing this gap requires a shift from tool-oriented instruction toward **computational BIM literacy**, where modeling, data quality, automation, and performance reasoning are treated as interconnected aspects of a single workflow.

While prior BIM education research emphasizes collaboration, interoperability, and process integration, the ex-

plicit integration of programming, API-based data access, and performance reasoning into core BIM teaching remains largely unaddressed.

This paper investigates the following research question: How can BIM education be redesigned so that students develop computational BIM literacy - understanding BIM models as data-driven, queryable, and analyzable artifacts - rather than learning BIM primarily as a geometric authoring process?

To address this question, we designed, implemented, and evaluated a project-based educational framework that integrates programming, API-based model access, and simplified multidisciplinary performance reasoning within a single workflow.

The core pedagogical mechanism is that simplified performance models foster a learning-by-doing environment that exposes BIM data quality issues during computation.

In this work, the term *framework* refers to a transferable, tool-agnostic structure that integrates (1) BIM data access via APIs, (2) computational logic implemented in Python, and (3) simplified multidisciplinary performance models. The framework has been applied successfully with Revit, ArchiCAD, and Vectorworks, demonstrating that it is not dependent on any specific authoring tool. While the competencies addressed-data structuring, scripting, API querying, and performance reasoning-are most relevant for architecture and engineering students.

Recent developments in BIM education research highlight that these challenges are not only technical but pedagogical. Studies emphasize the importance of constructivist and project-based learning environments for developing digital competencies, particularly when students interact with computational models and simplified analyses Ao et al. (2025b); Nguyen and Adhikari (2025). Positioning BIM teaching within such learning theories reinforces the need for hands-on, exploratory workflows that connect model semantics, assumptions, and analytical consequences. This perspective motivates the project-based computational framework presented in this work.

This paper further positions the framework in recent educational research, describes the course's overall concept and learning objectives, before detailing the computational BIM workflow. It explains the project-based learning and describes the early performance reasoning for the energy and static topics. We refer to the LLM-supported learning, present evaluation results, and finally conclude the paper with a discussion and outlook.

Positioning within Existing BIM Education Research

Recent educational approaches further emphasize the need to move beyond tool-centric BIM teaching toward model-centric, computational, and interdisciplinary learning environments. The concept of "Next Generation BIM" frames BIM models as active carriers of data, logic, and assumptions that support automation, early performance reasoning, and life-cycle thinking across disciplines. This perspective explicitly calls for integrating programming, data access, and computational workflows into BIM education rather than treating them as advanced or optional skills (Wimmer et al., 2025).

Research on BIM education consistently highlights a gap between industry demands and academic training. Early BIM frameworks focused on maturity levels, collaboration, and process integration, while noting that education often lags behind professional practice (Succar, 2009). More recent studies confirm that BIM is frequently taught as software proficiency rather than as an integrated methodological paradigm (Eastman et al., 2018).

Recent global reviews highlight persistent gaps in BIM education, particularly in understanding student learning pathways, interdisciplinary curriculum design, and assessment practices Ao et al. (2025a). Moreover, contemporary studies show that computational thinking, programming, and automation remain underrepresented in most BIM programs despite their increasing relevance in industry. Evidence from AI-supported classroom environments demonstrates that students frequently struggle with debugging and data schema interpretation, underscoring the need for computational literacy as a core educational objective Sahraoui et al. (2025).

Large-scale empirical work further reveals a perception–practice gap: students appreciate the value of BIM for sustainability and analytical workflows but often lack the practical ability to apply these concepts independently Nguyen and Adhikari (2025). This finding aligns with our observations that students rarely develop a deep awareness of model structure, data consistency, and semantic assumptions when BIM is taught only through tool-oriented modeling.

Pedagogically, our approach aligns with constructivist and project-based learning principles. Recent BIM education research highlights the importance of learning motivation, behavioral intention, and authentic project engagement in developing digital competencies Ao et al. (2025b). Global curriculum reports also indicate a shift towards experiential learning models that integrate AI, automation, and computational reasoning Li (2025); Sofiane (2024).

Model-centric and openBIM-oriented teaching approaches have improved interdisciplinary collaboration and awareness of data exchange. However, computational literacy and automation remain underrepresented in many curricula. Prior work on life-cycle-oriented BIM education emphasizes interoperability and collaboration but often relies on predefined workflows and tool-specific processes (Maile et al., 2023). Conceptual frameworks for open BIM education similarly identify limited integration of programming and computational reasoning (Preidel et al., 2024).

The framework presented in this paper builds on this body of work by explicitly integrating programming and API-based model access into core BIM education. Rather than treating computation as an add-on, BIM models are framed as computational artifacts that can be queried,

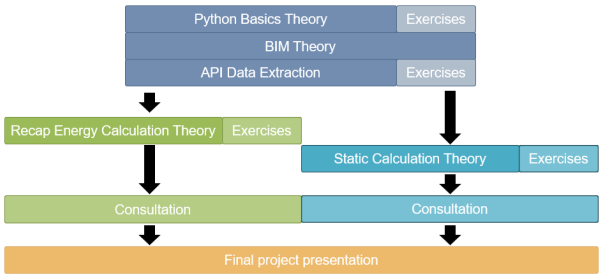

Figure 1: Phased course structure combining short theoretical inputs with exercises, consultations, and iterative project work. Programming, BIM theory, and performance reasoning are introduced incrementally and integrated into a single project-based workflow.

transformed, and evaluated across multiple performance domains.

Course Concept and Learning Objectives

The framework was implemented as a CAX module in the master program Energy Efficiency Design – E2D at the Technical University of Applied Sciences, Augsburg. Programming experience ranged from none to intermediate, and the course was explicitly designed to accommodate this diversity. Learning objectives focused on developing computational BIM literacy rather than tool proficiency. Students were expected to

1. understand BIM models as structured, data-centric representations,
2. access and query models via authoring APIs,
3. formulate explicit modeling and analysis assumptions,
4. implement transparent computational logic, and
5. interpret early-stage performance results critically.

Python serves as a lightweight orchestration layer, supporting analytical reasoning rather than being a primary goal of the work.

Course Structure. Teaching followed a phased progression, combining theoretical inputs with exercises and eventually project-based application as outlined in Figure 1. Programming fundamentals, BIM theory, and API-based data access for the first part of the course. Concepts of energy and structural performance reasoning were then introduced through accompanying exercises. The final part of the course was performing the project from modeling to code writing to result interpretation. In this phase, the students are supported through multiple consultation sessions to answer technical and scope questions and guide them towards a successful project. Consultation sessions provide targeted support and enable reflection on both modeling and computational decisions. Finally, the project was presented by each group in front of the class.

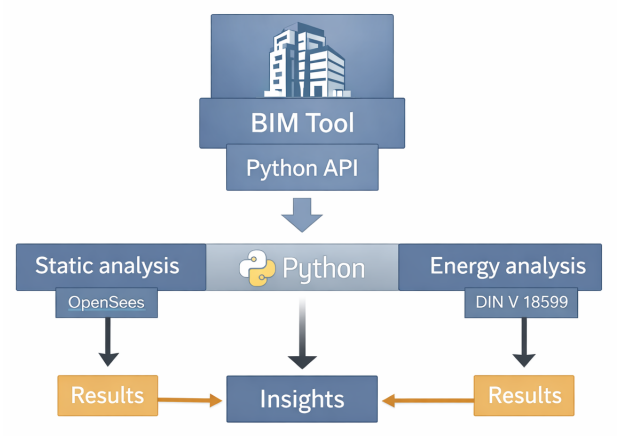

Figure 2: Computational BIM workflow used in the course. BIM models are accessed via authoring APIs and processed through Python-based computational logic to support parallel energy and structural reasoning and the generation of integrated insights.

Methods: Course Structure and Implementation

The course was conducted over 14 weeks (5 ECTS) and included students from the Energy Efficiency Design master program. The weekly structure consisted of typically 90-120 minutes of conceptual input (Python basics, BIM theory, energy and structural fundamentals) followed by guided hands-on sessions on API data access, code debugging, and computational workflow development.

Students were provided with Python templates, API documentation, example extraction routines, and simplified analytical formulations. Their final project required them to: (1) model a small building; (2) extract geometric and semantic data through the authoring tool's API; (3) compute simplified energy balances and generate structural models in OpenSees; and (4) present their assumptions, computational decisions, and debugging steps. Emphasis was placed on transparency, reproducibility, and reasoning rather than numerical accuracy.

Computational BIM Workflow

Students created BIM models using standard authoring tools. While most groups worked with Autodesk Revit, individual groups experimented with Vectorworks and ArchiCAD. The framework was intentionally designed to remain independent of specific authoring environments, focusing on API-based access to model data rather than tool-specific workflows.

Figure 2 summarizes the computational workflow used throughout the course. BIM geometry and metadata are accessed via authoring APIs and processed using Python scripts. Relevant information is mapped into simplified analytical representations for energy and structural reasoning, allowing students to explore performance implications of modeling decisions early in the design process.

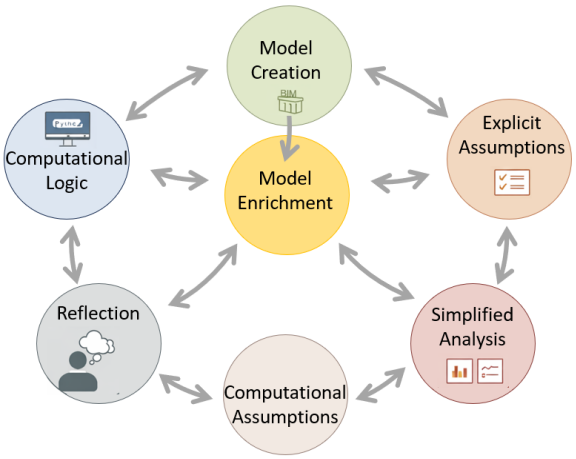

Figure 3: Learning-by-doing and reasoning loop adopted in the course. Students iteratively create BIM models, formulate explicit assumptions, implement computational logic, analyze simplified results, and reflect on outcomes to guide subsequent design iterations.

Object-Oriented Structuring as Didactic Principle

To support transparent and meaningful automation, students were encouraged to structure their scripts using basic object-oriented principles. Computational tasks were organized into domain-specific components, such as energy-related calculations, structural abstractions, and unit handling, rather than monolithic scripts.

This structuring was introduced as a didactic scaffold rather than formal software engineering training. Aligning code structure with domain concepts helped students clarify responsibilities, make assumptions explicit, improve the traceability of computational logic, and ease debugging and testing. In earlier course iterations, debugging within the Python API environment proved challenging for students and motivated the introduction of a more structured coding approach. This is why the object-oriented approach naturally emerged as a way to organize code and provide enough structure so that error checking and debugging were improved.

Human-in-the-loop computational assistance

Large language models were used as human-in-the-loop assistants to support syntax, debugging, and API exploration. Automated solution generation with LLMs was explicitly discouraged. Students remained responsible for validating logic and interpreting results. Recent work suggests that such assistive use of LLMs can effectively support exploratory reasoning in AEC contexts (Maile and Güldür Erkal, 2025).

Project-Based Learning Setup

The course followed an open-ended project-based learning approach without predefined solutions. Students were responsible for model creation, scripting, assumption definition, and interpretation of results.

Figure 3 illustrates the iterative learning process. Students repeatedly move between modeling, computation, and interpretation, enabling them to understand how design decisions, data quality, and assumptions influence analytical outcomes.

Early Performance Reasoning

The complementary expertise of the instructors enabled the integration of early performance reasoning in both energy and structural domains.

Energy Reasoning

Energy reasoning was introduced using a simplified monthly steady-state balance formulation that explicitly distinguishes between transmission losses, ventilation losses, internal gains, solar gains, and heating. This builds on prior coursework in construction physics in the related bachelor's program, where students were introduced to these principles.

This energy balance abstraction, used in the course, is based on the German Standard DIN V 18599 and is further simplified to omit storage effects and system efficiencies. Instead of aiming for regulatory compliance or high-fidelity simulation, the approach emphasizes transparency and causal understanding. Students explicitly define all assumptions and directly experience how changes in geometry, envelope properties, or usage profiles affect heating demand. For the various energy balance terms, the following information was needed from the BIM model:

- ventilation losses:
  - building volume
- transmission losses (for all building elements)
  - external surface area
  - u-value
- solar gains (for all windows)
  - g-value/solar heat gain coefficient (SHGC)
  - external surface area
  - orientation
- internal gains
  - usable area inside the building

Static Reasoning

Structural reasoning was introduced as a simplified approach to support early understanding of structural behavior within a CAX workflow. Rather than performing detailed structural design, the goal was to help students understand how structural layout and geometry influence global structural response.

The following key elements were obtained from the BIM model:

- nodal coordinates derived from element geometry,
- connectivity between structural elements,
- element types such as columns, beams, walls, and slabs,
- idealized material and cross-section properties,

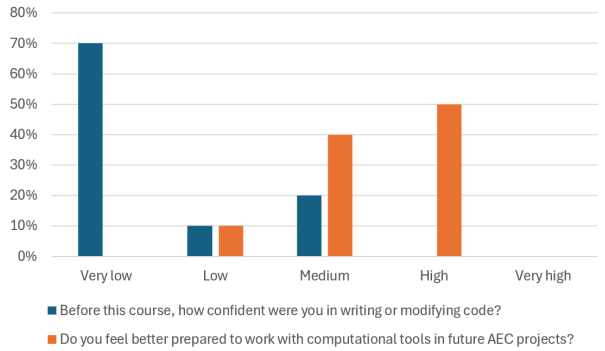

Figure 4: Self-assessed student confidence before and after the course. Responses indicate low initial confidence in writing or modifying code and a substantial increase in perceived preparedness to work with computational tools in future AEC projects.

- support conditions and boundary constraints.

Starting from the BIM model, students used authoring APIs to extract structural information and translate it into simplified analytical representations. Based on this extracted information, students automatically generated a code-based structural model in `Tcl` format for use in OpenSees (McKenna et al., 2010). Reduced assumptions appropriate for early design stages were applied to maintain transparency and ease of interpretation. This allowed students to directly observe how changes in BIM geometry or structural layout affected the generated OpenSees model and its global structural response.

The limitations of this simplified approach were explicitly addressed by emphasizing that the generated models represent idealized structural systems and provide insight into overall behavior rather than detailed effects.

Evaluation and Observed Learning Effects

Evaluation was based on qualitative and self-assessed feedback collected through an anonymous questionnaire at the end of the course, combined with evidence from student project presentations and submitted computational artifacts. In total, 23 students participated in the course, of which n=10 completed the questionnaire.

Questionnaire results. Students were asked to reflect on their prior experience with programming and their perceived preparedness to work with computational tools in future AEC projects. Figure 5 illustrates the distribution of responses. The results show that most students entered the course with low confidence in writing or modifying code, confirming the heterogeneous and largely non-programming-oriented background typical of AEC cohorts. Despite this, students reported a clear shift toward higher confidence in using computational tools after completing the course.

Students frequently emphasized that simplified models, explicit assumptions, and hands-on experimentation were more valuable for understanding performance relationships than black-box tools or lecture-only formats. Rather than focusing on numerical accuracy, students highlighted an improved ability to reason about cause–effect relationships and to explain why results changed.

Evidence from student work. Beyond self-assessment, student project outcomes provided concrete evidence of learning effects. Across all teams, students demonstrated mastery in three interconnected domains:

1. BIM modeling quality,
2. computational data extraction and analysis, and
3. integrated energy and structural reasoning.

Students developed a strong awareness that reliable downstream analyses depend on precise geometry, correct units, and well-structured parameter data. Several teams reported discovering unit inconsistencies (e.g. feet-to-meter conversions), missing thermal parameters, and incorrect element assignments only when attempting to compute energy balances or generate structural models from their BIM data. These experiences directly reinforced the importance of semantically consistent modeling practices. In multiple cases, students had to revise their BIM models because computational scripts exposed missing or inconsistent parameters.

All teams successfully implemented Python-based workflows that accessed BIM data via authoring APIs to compute simplified energy balances and generate analytical structural models for OpenSees. Students produced reproducible outputs such as CSV files, TCL scripts, and graphical plots derived directly from their BIM models. This demonstrates that students were not only able to model buildings but to compute with their models across multiple domains.

Several groups explicitly recognized interdependencies between energy performance and structural modeling. They observed how geometric decisions influenced both thermal behavior and structural response, for example how envelope area affects transmission losses while structural layout influences mass and connectivity. The use of a single parametric BIM model feeding both energy and structural analyses strengthened this integrated design understanding.

Typical outputs produced by the groups included CSV files listing extracted surface areas, thermal parameters, and orientations; Python-generated plots of monthly heating demand; and automatically generated Tcl files for simplified OpenSees models. Several teams performed gravity load analyses and modal evaluations, demonstrating how geometric changes in the BIM model directly influenced the analytical model.

Common errors included inconsistent units (e.g., feet-to-meters), missing or misassigned U-values, disconnected analytical nodes in slabs, incorrect window orientation extraction, and Python path issues in the Revit environment. These issues only became visible during computation, reinforcing the pedagogical principle that data quality awareness emerges most effectively when students compute with their own model data.

Table 1: Summary of lessons learned across all student teams.

Competency Area	Lessons Learned (Evidence)
BIM Model Quality	Students emphasized precise geometry, correct units, and complete thermal and structural parameters. Errors such as incorrect unit conversions reinforced the need for careful model structuring.
Data Extraction and API Usage	Students extracted areas, volumes, U-values, materials, and orientations using authoring APIs. They automated energy balance calculations and implemented robust parameter detection.
Energy Calculation Concepts	Teams implemented simplified energy balances including transmission and ventilation losses, internal and solar gains, and heating demand, recognizing how geometry and assumptions influence results.
Structural Modeling and Analysis	Students exported analytical models and executed OpenSees simulations. They learned the importance of node alignment, connectivity, and consistent element definitions.
Integrated Design Thinking	Students identified synergies and trade-offs between energy demand and structural utilization. A single BIM model fed both energy and structural workflows, enabling informed design decisions.
Workflow and Project Management	Students emphasized consistent naming, version control, file organization, and avoiding special-character issues in BIM workflows.
Debugging and Problem Solving	Students encountered and resolved API syntax errors, incorrect file paths, misassigned properties, and parameter inconsistencies, reinforcing reflective learning.

Students also reported meta-level workflow lessons, including the importance of consistent naming conventions, file organization, API error handling, and validation of modeling assumptions. These reflections indicate that the course fostered realistic digital engineering competencies beyond isolated tool usage.

Table 1 summarizes the lessons learned across all student teams.

Observed learning effects. Three dominant learning effects were observed across participants (as shown in Table 2):

- *Increased programming confidence.* Students reported improved understanding of basic programming concepts such as variables, loops, and functions, indicating a reduced computational entry barrier.
- *Shift in BIM understanding.* Several students described a conceptual shift from geometry-focused modeling toward working with structured, queryable data.
- *Improved performance reasoning.* Implementing simplified energy and structural calculations improved students' confidence in interpreting results rather than accepting outputs from black-box tools.

The role of large language models was also highlighted by many students. LLM tools were primarily used for syntax support, debugging, and API exploration, enabling students to focus on conceptual reasoning.

Overall, the triangulated evidence from questionnaires, student artifacts, and project presentations indicates that treating BIM models as computational artifacts - combined with learning-by-doing and transparent simplifications - can foster meaningful computational BIM literacy even among students with limited prior coding experience.

Implementation Challenges

Integrating programming into BIM education required a substantial time investment, particularly for students with limited coding experience. Approximately one-third of weekly contact time was dedicated to debugging assistance and API exploration. These observations mirror findings from recent studies showing that students in AI-assisted BIM classrooms struggle with syntax, rule formulation, and data-schema interpretation Sahraoui et al. (2025). Instructor expertise in both BIM semantics and computational methods was essential to ensure that the cognitive load remained manageable within a project-based learning environment.

Table 2: Summary of qualitative student feedback (self-reported).

Dimension	Dominant Observation
Prior programming experience	Predominantly low or very low
Perceived programming understanding after course	Improved or partially improved
BIM understanding shift	From geometry-focused to data-oriented (for most students)
Usefulness of LLM assistance	High for syntax, debugging, and API exploration
Performance reasoning confidence	Improved understanding and interpretability
Ability to extend scripts independently	Possible with guidance; limited full independence

Limitations

The evaluation is qualitative in nature and based on a single course iteration with a limited number of participants. No control group was used, and results reflect perceived learning outcomes rather than measured performance gains. Future work will include longitudinal and comparative studies.

Discussion and Outlook

The presented framework addresses several persistent shortcomings in BIM education, including tool-centric teaching, black-box workflows, and fragmented multidisciplinary analysis. By treating BIM models as computational artifacts rather than geometric authoring results, the course enabled students to directly experience how data quality, modeling assumptions, and semantic consistency influence analytical outcomes.

A key insight emerging from this framework is that simplified performance models act as a didactic mirror. When students attempt to compute energy demand or generate structural models from their BIM data, inconsistencies, missing parameters, unit errors, and modeling assumptions become immediately visible. Rather than discovering such issues in later project phases or external tools, students encounter them during the act of computation. This immediate feedback loop fosters a causal understanding of how modeling decisions propagate through digital workflows and strengthens awareness of data quality and consistency.

This approach differs fundamentally from conventional BIM teaching practices. Conventional teaching often follows a sequence of modeling, exporting, and processing models in domain-specific tools. In contrast, the presented framework establishes a direct connection between the BIM model and computational reasoning through API access and scripting. This shifts the educational focus, as shown in Table 3:

Table 3: Comparison between traditional BIM education and the presented computational framework.

Traditional BIM education	Presented framework
Geometry-focused modeling	Data- and structure-focused modeling
Tool proficiency	Computational BIM literacy
Predefined workflows	Querying and transforming model data
Black-box simulations	Transparent simplified reasoning
Fragmented domain analyses	Multiple analyses from the same model

The results indicate that even students with little or no programming experience can develop meaningful computational literacy when programming is framed as a reasoning tool rather than an end in itself. The use of Python as a lightweight orchestration language, combined with object-oriented structuring aligned to domain concepts, allowed students to organize computational logic in a way that was traceable and understandable.

The integration of large language models as human-in-the-loop assistants, enabling students to make and implement their own decisions, further supported this process. Students reported that LLM tools helped them overcome syntactic barriers and explore APIs, allowing them to focus on conceptual reasoning. Importantly, responsibility for validating logic and interpreting results remained with the students, reinforcing critical engagement rather than automated solution generation.

Although the evaluation is qualitative and limited to a single course iteration, observed learning effects and student feedback indicate a shift in how BIM is perceived - from a geometric modeling task to working with structured data that can be queried, transformed, and analyzed. Students reported improved confidence in programming, a changed understanding of BIM, and an increased ability to reason about performance outcomes rather than accept results from black-box tools.

A central strength of the framework lies in its transferability. The approach does not depend on specific authoring software but on the availability of an API and basic scripting capabilities. This makes the framework adaptable to

different BIM environments and educational contexts. Future investigations into open-source BIM platforms and API-rich environments may further enhance its accessibility.

Future work will extend the framework to additional performance domains such as embodied carbon assessment and cost analysis, enabling students to explore further dimensions of model-based reasoning. Longitudinal studies and comparative evaluations with traditional BIM courses are planned to assess learning effects quantitatively. The integration of agent-based and LLM-supported workflows also represents a promising direction for further research into computational BIM education. Future work will explore integrating the Model Context Protocol (MCP) to enable LLMs to access and query BIM models through standardized, auditable tool interfaces (MCP, 2026; Anthropic, 2024). First preliminary investigations of this approach promise more transparent, reproducible, and educationally effective natural-language interactions with BIM data by explicitly linking user intent to authoring-tool API operations.

Ultimately, this work suggests that BIM education can move beyond teaching how to model buildings toward teaching how to compute with building models. By embedding programming, API access, and simplified performance reasoning into core BIM teaching, students can develop the computational literacy necessary to understand, question, and extend digital workflows in contemporary AEC practice.

Beyond its immediate didactic benefits, the framework contributes to broader discussions in BIM education. Recent reviews call for integrated, interdisciplinary, and computationally grounded learning models Ao et al. (2025a); Li (2025). Our findings support these directions by demonstrating that constructivist, project-based environments, combined with transparent computational workflows, can reduce the perception–practice gap documented in current educational research Nguyen and Adhikari (2025). By embedding programming and performance reasoning directly into students' modeling workflows, the course advances emerging pedagogical aspirations for computational BIM literacy.

Acknowledgments

We want to thank the students of the CAX module for their participation in the course and for their feedback.

References

Anthropic (2024). Introducing the model context protocol. `https://www.anthropic.com/news/model-context-protocol`. Accessed Feb 8, 2026.

Ao, Y., Wang, Y., Peng, P., and Zhao, J. (2025a). Bim adoption: A review of global trends, research progress, and future directions. In Building Information Modeling (BIM) Adoption, pages 1–12. Springer.

Ao, Y., Wang, Y., Peng, P., and Zhao, J. (2025b). Conclusion and future research: Bridging bim education, engagement, and industry readiness. In Building Information Modeling (BIM) Adoption, pages 163–169. Springer.

Borrmann, A., König, M., Koch, C., and Beetz, J., editors (2018). Building Information Modeling: Technology Foundations and Industry Practice. Springer.

Eastman, C., Teicholz, P., Sacks, R., and Lee, G. (2018). BIM Handbook. Wiley, 3 edition.

Li, K. (2025). Bim education – global 2025 update report. Technical report, NATSPEC. Version 12.0.

Maile, T., Bartels, N., and Wimmer, R. (2023). Integrated life-cycle oriented teaching of the big-open-bim method. In Proceedings of the 2023 European Conference on Computing in Construction (EC3) and CIB W78 Conference.

Maile, T. and Güldür Erkal, B. (2025). Ai-driven insights in aec: Literature discovery and practical applications. In Proceedings of the 2025 European Conference on Computing in Construction (EC3) and CIB W78 Conference, Porto, Portugal.

McKenna, F., Fenves, G. L., and Scott, M. H. (2010). OpenSees: Open system for earthquake engineering simulation. `https://opensees.berkeley.edu`.

MCP (2026). Model context protocol — official documentation. `https://modelcontextprotocol.io/`. Accessed Feb 8, 2026.

Nguyen, T. D. and Adhikari, S. (2025). Bridging the gap: Enhancing bim education for sustainable design through integrated curriculum and student perception analysis. Computers, 14(11):463.

Preidel, C., Bartels, N., Maile, T., and Wimmer, R. (2024). An open platform for building information modeling education: Conceptual framework and potential impact. In Proceedings of the CIB W78 Conference.

Sahraoui, I., Kim, K., Gao, L., Din, Z., and Senouci, A. (2025). Integrating generative ai in bim education: Insights from classroom implementation. arXiv.

Sofiane, M. (2024). The 2024 natspec report: A detailed overview of bim education and maturity.

Succar, B. (2009). Building information modelling framework: A research and delivery foundation for industry stakeholders. Automation in Construction, 18(3):357–375.

Wimmer, R., Bartels, N., and Maile, T., editors (2025). Next Generation BIM: Für Praxis und Lehre. bSD Verlag.