BIMARENA: BENCHMARKING DOMAIN-SPECIALIZED COMPUTER-USE IN PROFESSIONAL BUILDING MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Computer-use agents are advancing rapidly; yet, existing benchmarks focus on general-purpose applications and do not fully reflect the complexity of real-world workflows. Moreover, they tend to prioritize breadth across applications over depth within professional workflows that span diverse functionalities and tightly coupled operations. In this work, we introduce BIMArena, a domain-specialized benchmark for evaluating the depth of computer-use agent capabilities in Building Information Modeling (BIM) authoring environments, a complex, knowledge-intensive professional domain that remains underexplored in current benchmarks. BIMArena comprises 409 tasks that progressively evaluate fundamental professional software operations, long-sequence workflows, and domain reasoning tasks. We further investigate whether external operational support through software documentation and expert-curated operational skills improves agent performance. Experiments are conducted in isolated real-computer environments using a programmatic evaluation framework. They reveal a pronounced capability gap: the strongest agent achieves a 98.4% success rate on fundamental software operations, yet only 26.6% on realistic, professional tasks. Our analysis shows that models strong on general computer-use benchmarks can still struggle with professional tasks beyond basic user interface operations. Operational support also becomes less effective with increasing task depth and can even be detrimental.

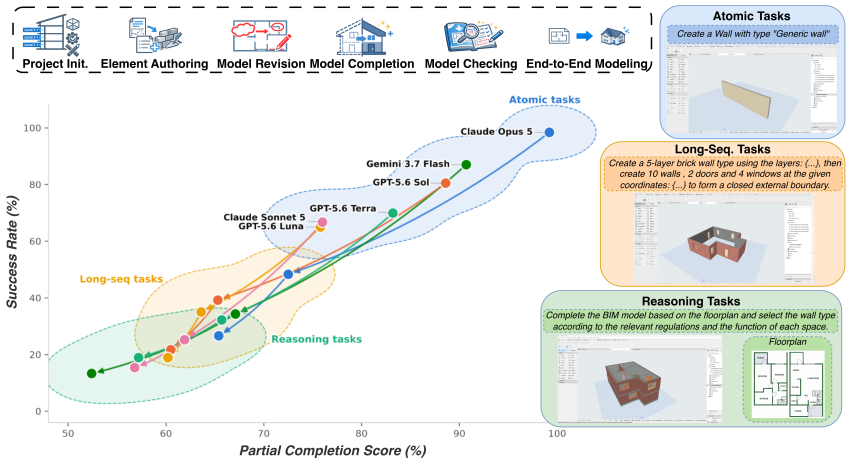


Figure 1: BIMArena comprises 409 building modeling tasks across levels: Atomic, Long-Sequence, and Reasoning. Computer-use agent performance declines sharply as task depth increases.

1 INTRODUCTION

Graphical User Interface (GUI) computer-use agents offer a promising way to interact with computers in a "human-like" manner through mouse and keyboard actions, enabled by recent advances

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

in large multimodal models (LMMs) (Anthropic, 2024; OpenAI, 2025; Xu et al., 2025b; Agashe et al., 2025a; Qin et al., 2025; Wang et al., 2025; Liu et al., 2026). Building on this progress, agents increasingly exhibit competencies in visual perception, deliberative planning, and GUI action execution (Cheng et al., 2024; Li et al., 2025b; Hu et al., 2025; Wei et al., 2026). Accordingly, a growing body of benchmarks has emerged to assess these capabilities across diverse tasks (Deng et al., 2023; Xie et al., 2024; Xu et al., 2025a; Zhou et al., 2024; Bonatti et al., 2025).

With the increasing capability of computer-use agents, the scope of computer-use agents is gradually expanding from rudimentary tasks to more sophisticated and specialized domains, such as gaming (Tan et al., 2025b; Ahn et al., 2025), science (Sun et al., 2025; 2026), and engineering (Man et al., 2025; Ai et al., 2026b; Deng et al., 2025). However, evaluating agents in these specialized settings becomes substantially more challenging due to (1) complex and implicit software interactions, (2) long-horizon workflows, and (3) the required knowledge that governs the appropriate sequence of operations. Several recent benchmarks have begun to address these challenges (Yuan et al., 2026; Ai et al., 2026a; Sun et al., 2026). Nevertheless, they remain breadth-oriented, prioritizing coverage across diverse applications and domains while underexploring the rich functional diversity and heterogeneous workflows that exist within a single specialized professional domain.

To address this gap, we introduce BIMArena, a domain-specialized computer-use benchmark that evaluates agents across diverse functionalities and realistic workflow stages within a single domain: Building Information Modeling (BIM) (Borrmann et al., 2018). BIM provides a well-suited testbed for domain-specialized evaluation, as its information-rich digital building representations give rise to sophisticated authoring workflows characterized by complex user interfaces and substantial domain-knowledge requirements (Du et al., 2024; 2025; Park et al., 2026). As tasks become more complex, agents must handle not only longer workflows but also richer domain information across interdependent decisions. We therefore use depth to capture both dimensions. BIMArena comprises 409 tasks across two widely used BIM authoring software platforms, organized into a progressive three-level hierarchy to emphasize domain depth: *Atomic* (123 tasks) assesses fundamental professional software operations; *Long-Sequence* (143 tasks) evaluates realistic long-horizon workflows under explicit task specifications; and *Reasoning* (143 tasks) additionally requires domain-specialized reasoning before workflow execution. Tasks are meticulously designed and constructed by domain experts to cover a wide range of capabilities required in practice, including building model understanding, model creation and modification, architectural drawing interpretation, and end-to-end modeling workflows. To ensure reliable evaluation, all tasks are executed in isolated virtual environments with a programmatic, state-based evaluation pipeline. Beyond task success, BIMArena also measures partial completion and unintended modifications to the existing model. We further study whether external operational support from official documentation and expert-derived operational skills can improve agent performance in professional software environments.

The findings reveal substantial performance disparities across models, even in fundamental software operations. The strongest model achieves success rates of 98.4% with operational support and 95.9% without support, whereas the weakest reaches only 4.9% with support and 2.4% without support. More importantly, strong competence in fundamental software operations does not translate to deeper professional workflows: the strongest model drops by 50.1 percentage points from Atomic to Long-Sequence tasks and by 71.8 percentage points from Atomic to Reasoning tasks. Our analysis shows that successful cases remain concentrated on UI interaction, whereas failures increasingly arise from grounding, understanding, and manipulating building elements. External operational support only partially mitigates this gap, with diminishing benefits as task complexity increases. Its effectiveness further depends on retrieval precision, as less relevant information can introduce additional distractions rather than useful guidance. Taken together, they reveal a clear domain-specialized capability gap in GUI computer-use agents.

2 Related work

2.1 Computer-use Agents

Given user instructions, computer-use agents execute tasks by interacting directly with operating systems and applications (Zhang et al., 2025; Hu et al., 2025). Depending on the interaction modality, two common paradigms are Command-Line Interface (CLI)-based and GUI-based agents. CLI agents interact with systems by issuing textual commands or generating executable scripts for direct

system-level control (Wang et al., 2024; Wu et al., 2024; Yang et al., 2026), whereas GUI agents perceive visual interfaces and operate applications directly through mouse and keyboard actions (Nguyen et al., 2025; Agashe et al., 2025b; Qin et al., 2025; Wang et al., 2025). Recently, these paradigms have been extended beyond general-purpose computer tasks to increasingly complex and specialized application domains (Tan et al., 2025b;a; Man et al., 2025; Sun et al., 2025; Li et al., 2025a; Fu et al., 2025). However, evaluation across these specialized settings remains fragmented and provides limited insight into agent capabilities within complex professional workflows.

2.2 Benchmark for Computer-use Agents

Early computer-use benchmarks primarily focused on interface grounding, evaluating whether agents could identify and interact with the correct UI elements (Deng et al., 2023; Kapoor et al., 2024; Li et al., 2025b). More recent benchmarks have shifted toward dynamic settings in which agents repeatedly observe, act, and receive updated states, with success determined by the final environment state (Xie et al., 2024; Zhou et al., 2024; Rawles et al., 2025; Koh et al., 2024; Bonatti et al., 2025). This paradigm has further expanded toward longer and realistic workflows (Yuan et al., 2026; Wang et al., 2026a) and alternative interaction modalities (Jia et al., 2026; Zhou et al., 2026; Wang et al., 2026b). In terms of scope, recent benchmarks have also begun incorporating professional software (Yuan et al., 2026; Wang et al., 2026a; Ai et al., 2026a; Sun et al., 2026). However, their emphasis on broad coverage across applications and domains often comes at the expense of depth within any single professional domain. Domain-specialized engineering benchmarks such as ComCADBench (Ai et al., 2026b), BMEval (Jiang et al., 2026), and BIM-Edit (Nithyanantham et al., 2026) provide deeper task coverage but primarily formulate tasks as programmatic generation or structured model manipulation. Consequently, domain-specialized computer use involving long-horizon execution in native professional software remains largely underexplored.

3 BIMArena

In this section, we introduce the progressive task design and construction pipeline shown in Figure 2, followed by benchmark statistics that demonstrate broad coverage across BIM authoring workflows.

3.1 Building Modeling Fundamental Capability Taxonomy

BIM authoring tasks performed within professional software span a wide range of operations, from creating and deleting building elements to modifying existing models. Although platforms differ substantially in interface design and software-specific workflows, they share a largely overlapping set of underlying modeling capabilities. To capture these common capabilities, two domain experts reviewed recurring BIM authoring operations across official software documentation and tutorials, then consolidated functionally equivalent operations into software-independent capability definitions based on their practical BIM authoring experience. In this way, we identify 23 fundamental capabilities as reusable functional primitives that specify **what** needs to be accomplished, independent of how a particular software application realizes each operation. The complete capability taxonomy is provided in Appendix A.3.

3.2 Progressive Task Design and Construction

Successfully completing real-world BIM authoring workflows requires progressively deeper capabilities: (1) **Atomic**: reliable interactions with complex professional software interfaces; (2) **Long-Sequence**: long-horizon planning and execution across realistic building modeling tasks; and (3) **Reasoning**: domain-grounded interpretation of drawings, specifications, and other contextual information to determine appropriate modeling decisions.

Expert-Guided Task Construction. The benchmark tasks are constructed by five BIM experts, each with at least two years of experience in BIM authoring using the target applications. For Atomic tasks, the experts instantiate each capability into one or more variants by varying target elements, parameter settings, and modeling contexts while keeping the required interaction sequences relatively short. The experts then identify representative domain-specialized inputs (architectural drawings, minimum design requirements, and regulatory requirements) from architectural handbooks and

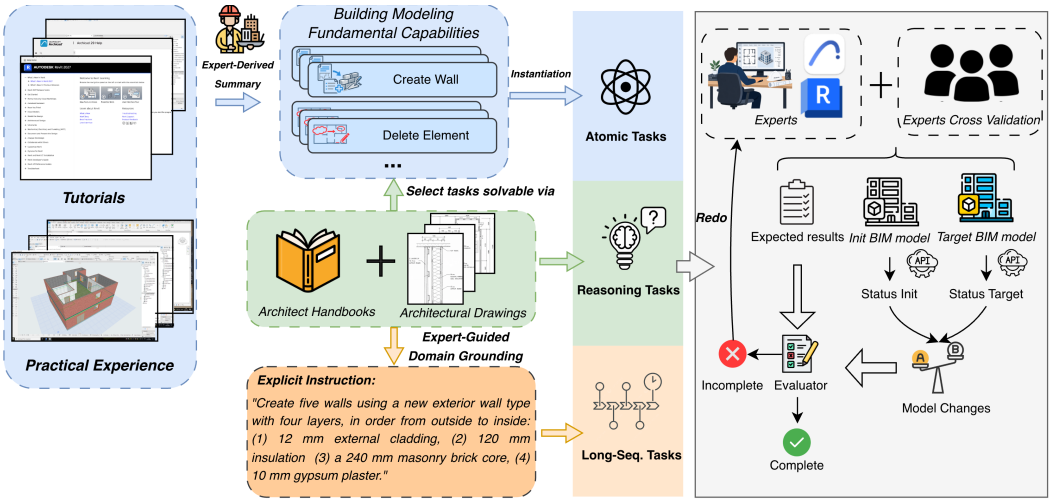


Figure 2: Overview of the task design, construction, and validation pipeline in BIMArena.

real-world practice while ensuring that the required software operations remain within the scope of the defined capability taxonomy. These inputs are used to construct the Reasoning tasks, where the modeling objective must be inferred from the provided context and relevant domain knowledge not explicitly included in the task description. Drawing on their domain expertise, the experts then explicitly formulate the corresponding modeling requirements in textual form to create the paired Long-Sequence tasks with the same underlying target state. Each task is further annotated with the target software and fixed software version, the initial BIM model, the target BIM model, and the task-specific evaluation constraints. Detailed task coverage, reference resources, task annotations, and representative task examples are provided in the Appendix A.

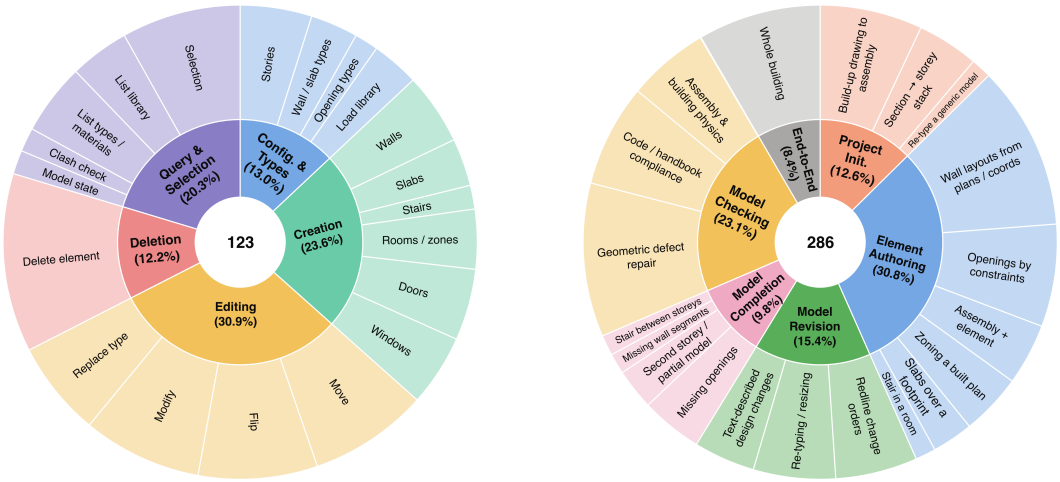
Quality Control. All tasks are cross-reviewed by domain experts. We perform script-based validation for each task: the expert-constructed target model S^* must satisfy all task-specific evaluation constraints, whereas the initial model S_0 must not already satisfy the set of evaluation criteria. Tasks that fail either check are revised and revalidated before inclusion. Details of the script-based validation are provided in Appendix C.4 and C.5.

3.3 BIMArena Statistics

Figure 3 shows the task distribution of BIMArena. The 123 Atomic tasks focus on fundamental software operations, whereas the 286 Long-Sequence and Reasoning tasks cover more complex tasks spanning the broader BIM authoring workflow. The statistics of BIMArena are summarized in Table 1. In total, BIMArena includes 409 tasks across Revit (Autodesk, 2026a) and Archicad (Graphisoft, 2025a), with 205 and 204 tasks, respectively. Atomic tasks require an average of 8.2 GUI steps from experts, compared with 69.2 GUI steps for Long-Sequence and Reasoning tasks. The latter involve an average of 3.5 distinct capabilities and 10.5 capability occurrences per task. The tasks set for the two software environments are constructed with closely matched task categories and workflow distributions. The comparison with existing benchmarks is provided in Appendix A.6.

Table 1: Statistics of BIMArena across the three evaluation tracks. Capability and GUI-step statistics are reported as average / minimum / maximum per task.

Statistic	Atomic	Long-Seq.	Reasoning	Overall
# Tasks	123 (30.1%)	143 (35.0%)	143 (35.0%)	409 (100%)
Archicad / Revit	62 / 61	71 / 72	71 / 72	204 / 205
Required capability types per task	1.0 / 1.0 / 1.0	3.5 / 2.0 / 9.0	3.5 / 2.0 / 9.0	2.8 / 1.0 / 9.0
Required capability instances per task	1.0 / 1.0 / 1.0	10.5 / 2.0 / 68.0	10.5 / 2.0 / 68.0	7.6 / 1.0 / 68.0
GUI steps per task	8.2 / 3.0 / 26.0	69.2 / 16.0 / 203.0	69.2 / 16.0 / 203.0	50.9 / 3.0 / 203.0



(a) Atomic tasks.

(b) Long-Sequence and Reasoning tasks.

Figure 3: Task distribution across BIMArena.

4 Experiments

4.1 Agent Framework

We use a lightweight agentic framework with screenshot-based interaction. At each step t, the agent receives the task instruction, the current and previous screenshots, and a compact interaction history containing prior reasoning and actions, and predicts the next action as $a_t = M(I_t, I_{t-1}, H_t, q)$. Here, I_t and I_{t-1} are the current and previous screenshots; H_t is the interaction history up to step $t-1$; q is the task instruction; M is the evaluated model. The action space consists of mouse and keyboard interactions. The complete action space is described in Appendix C.3.

4.2 External Operational Support

BIM authoring functions involve interaction procedures that are not directly apparent from the interface and often require consulting official documentation or tutorials. We investigate whether external software operational support can improve agent performance by providing optional access to two complementary sources: operational skills and official software usage documentation. Inspired by reusable agent Skills (Anthropic, 2026c) and workflow recording for skill construction (Microsoft, 2026), domain experts construct 46 software-specific GUI operational skills by developing one skill for each of the 23 fundamental BIM capabilities in both Revit and Archicad. Additionally, we collect and index the official documentation for both platforms. The two sources are exposed through separate retrieval tools, and the agent independently decides which tool to invoke as needed during execution. We compare two experimental settings: without support and with support. A detailed illustration of the operational support mechanism and examples is provided in Appendix B.

4.3 Execution environment

As shown in Figure 4, the evaluation infrastructure consists of two main components: isolated Windows 11 virtual machines (VMs) and the host machine. The VMs are initialized from a common golden image containing both BIM authoring applications. A VM manager controls the VM lifecycle, including initialization, snapshot creation, restoration, and state management. For each task, all GUI interactions are executed inside the VM through the target BIM authoring application. The benchmark code and results reside on the host.

Before evaluation, the VM manager configures the software environment through predefined configuration files. This includes the required API connections and application settings. A separate reusable snapshot is created for each BIM application, with the corresponding application running in an idle and configured state. Before each task, the VM is restored to this snapshot to ensure re-

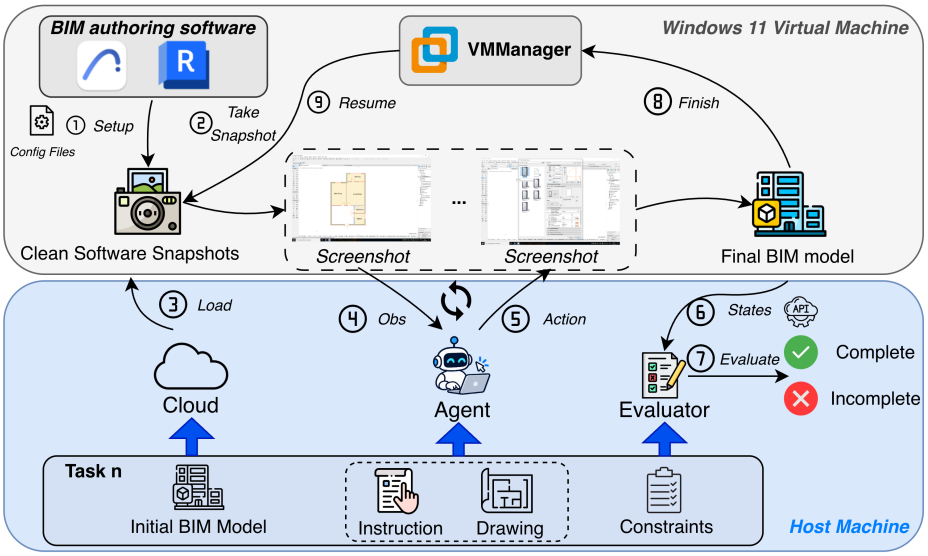


Figure 4: Overview of the BIMArena evaluation infrastructure, including VM-based task execution, agent interaction, and programmatic evaluation.

producible initialization. The task-specific initial BIM model is then transferred from the host-side to the VM and loaded into the corresponding application. During execution, the agent interacts with the VM in a loop. It receives the current VM screenshot as a visual observation and predicts the next action. The corresponding mouse or keyboard interaction is then executed in the VM. This process continues until the task terminates. The final BIM model state is extracted through the preconfigured software API and transmitted to the evaluator for programmatic evaluation. Once evaluation is completed, the VM manager restores the VM to the original snapshot and prepares it for the next task. Detailed environment configurations and software settings are provided in Appendix C.

4.4 Evaluation Metrics

For each benchmark task, domain experts define a set of task-specific evaluation constraints $\mathcal{C}^* = \{c_1, \ldots, c_n\}$, specifying the conditions that must hold in the final BIM model. After task execution, the resulting BIM model S_T is inspected through the native software APIs, and each criterion c_i is programmatically evaluated against the extracted model state. For a BIM authoring task, successful completion requires not only creating or modifying the requested elements correctly but also preserving existing elements that should remain unaffected. We therefore report the **Partial Completion Score (PCS)** to measure task progress. PCS is defined as the fraction of task-specific goal constraints satisfied by the final model. We additionally report the **Collateral-Free Rate (CFR)**, which measures the share of tasks where all preservation constraints are satisfied and captures whether unintended elements or properties remain unaffected. Based on these two criteria, we report **Success Rate (SR)** as the primary metric. A task is considered successful only when all goal constraints and all preservation constraints are satisfied, corresponding to PCS = 100% and CFR = 100%. Further details are provided in Appendix C.4 and C.5.

4.5 Models and Execution Settings

We evaluate nine state-of-the-art LMMs: GPT-5.6 Sol (OpenAI, 2026b), GPT-5.6 Terra (OpenAI, 2026c), GPT-5.6 Luna (OpenAI, 2026a), Claude Opus 5 (Anthropic, 2026a), Claude Sonnet 5 (Anthropic, 2026b), Gemini 3.7 Flash (Google DeepMind, 2026), Qwen3.8-Max (Alibaba Cloud, 2026b), Qwen3.7-Plus (Alibaba Cloud, 2026a), and Muse Spark 1.1 (Meta AI, 2026). We additionally evaluate two open-source computer-use agents: EvoCUA-32B (Xue et al., 2026) and OpenCUA-32B (Wang et al., 2025). Each task is assigned a maximum interaction budget measured by the number of agent rounds. Atomic tasks are limited to 100 rounds, while Long-Sequence and Reasoning tasks are limited to 300 rounds. All tasks are subject to a maximum wall-clock time of

60 minutes. Within each interaction round, the agent may invoke one or more actions (GUI steps or retrieval tools) from the available action space or terminate the task. EvoCUA-32B and OpenCUA-32B are evaluated only without operational support, as their native action interfaces do not support the retrieval tools required for this setting. Detailed model settings are provided in Appendix D.1.

Two-Stage Evaluation. We use the Atomic tasks as a screening stage before conducting the more expensive evaluation. While Atomic performance does not directly measure planning ability, reliable execution of fundamental capabilities is a prerequisite for success on long-horizon tasks. All models are therefore first evaluated on the Atomic tasks, after which the models achieving success rates above 50% are selected for full evaluation on the Long-Sequence and Reasoning tasks. This focuses the more expensive downstream evaluation on the most capable models while keeping the overall experimental cost tractable. Across all experiments, we estimate the total experimental cost at approximately USD 12,000. Detailed cost analysis is provided in Appendix D.2.

Table 2: Results of BIMArena, pooled over Archicad and Revit. SR = success rate (%); PCS = partial completion score (%); CFR = collateral-free rate (%). *w/* and *w/o* denote evaluation with and without support, respectively.

		Atomic			**Long-Seq.**			**Reasoning**			**Overall**
Model	**Support**	SR	PCS	CFR	SR	PCS	CFR	SR	PCS	CFR	SR
Claude Opus 5	w/o	95.9	97.6	97.6	46.9	70.7	70.6	**28.0**	63.3	70.6	55.0
	w/	**98.4**$^{+2.5}$	**99.2**$^{+1.6}$	**98.4**$^{+0.8}$	**48.3**$^{+1.4}$	**72.5**$^{+1.8}$	**74.8**$^{+4.2}$	26.6$^{-1.4}$	**65.4**$^{+2.1}$	69.2$^{-1.4}$	**55.7**$^{+0.7}$
Gemini 3.7 Flash	w/o	79.7	87.8	95.9	30.1	63.3	68.5	13.3	49.4	59.4	39.1
	w/	87.0$^{+7.3}$	90.7$^{+2.9}$	97.6$^{+1.7}$	34.3$^{+4.2}$	67.1$^{+3.8}$	72.7$^{+4.2}$	13.3$^{\pm 0.0}$	52.4$^{+3.0}$	65.0$^{+5.6}$	42.8$^{+3.7}$
GPT-5.6 Sol	w/o	69.9	78.5	91.1	31.5	62.3	66.4	17.5	58.2	58.7	38.1
	w/	80.5$^{+10.6}$	88.6$^{+10.1}$	96.7$^{+5.6}$	39.2$^{+7.7}$	65.3$^{+3.0}$	73.4$^{+7.0}$	21.7$^{+4.2}$	60.5$^{+2.3}$	70.6$^{+11.9}$	45.5$^{+7.4}$
GPT-5.6 Terra	w/o	60.2	74.9	88.6	27.3	54.9	65.7	16.1	51.5	64.3	33.3
	w/	69.9$^{+9.7}$	83.2$^{+8.3}$	90.2$^{+1.6}$	32.2$^{+4.9}$	65.7$^{+10.8}$	70.6$^{+4.9}$	18.9$^{+2.8}$	57.2$^{+5.7}$	66.4$^{+2.1}$	38.9$^{+5.6}$
GPT-5.6 Luna	w/o	58.5	71.2	92.7	24.5	58.1	59.4	18.9	53.7	67.1	32.8
	w/	65.0$^{+6.5}$	75.8$^{+4.6}$	91.9$^{-0.8}$	35.0$^{+10.5}$	63.6$^{+5.5}$	67.8$^{+8.4}$	18.9$^{\pm 0.0}$	60.2$^{+6.5}$	**71.3**$^{+4.2}$	38.4$^{+5.6}$
Claude Sonnet 5	w/o	58.5	75.6	87.8	25.2	56.1	63.6	16.1	50.6	61.5	32.0
	w/	66.7$^{+8.2}$	76.0$^{+0.4}$	92.7$^{+4.9}$	25.2$^{\pm 0.0}$	61.9$^{+5.8}$	70.6$^{+7.0}$	15.4$^{-0.7}$	56.8$^{+6.2}$	65.7$^{+4.2}$	34.2$^{+2.2}$
Avg. $\Delta_{w/-w/o}$ (full-track)	–	**+7.5**	**+4.7**	**+2.3**	**+4.8**	**+5.1**	**+6.0**	**+0.8**	**+4.3**	**+4.4**	**+4.2**
Qwen3.8-Max	w/o	7.3	12.6	81.3	–	–	–	–	–	–	7.3
	w/	10.6$^{+3.3}$	14.9$^{+2.3}$	82.9$^{+1.6}$	–	–	–	–	–	–	10.6$^{+3.3}$
Qwen3.7-Plus	w/o	2.4	7.8	78.9	–	–	–	–	–	–	2.4
	w/	4.9$^{+2.5}$	10.5$^{+2.7}$	79.7$^{+0.8}$	–	–	–	–	–	–	4.9$^{+2.5}$
Muse Spark 1.1	w/o	3.3	9.5	80.5	–	–	–	–	–	–	3.3
	w/	4.9$^{+1.6}$	11.9$^{+2.4}$	82.1$^{+1.6}$	–	–	–	–	–	–	4.9$^{+1.6}$
EvoCUA-32B	w/o	6.5	13.4	84.6	–	–	–	–	–	–	6.5
OpenCUA-32B	w/o	2.4	8.7	77.2	–	–	–	–	–	–	2.4

5 Results

5.1 Performance and Findings

As shown in Table 2, we compare the performance of nine advanced LMMs and two open-source computer-use agents on BIMArena. The main findings are summarized as follows:

Performance Gaps Across Evaluated Agents in Professional Software. We observe substantial variation in fundamental professional software competence across models. Claude Opus 5 achieves a 95.9% success rate and a 97.6% partial completion score on Atomic tasks without operational support. In contrast, OpenCUA-32B and Qwen3.7-Plus achieve task success rates of only 2.4%, with partial completion scores of around 8%; moreover, Qwen3.7-Plus benefits only marginally from operational support. These models have reported strong performance for general desktop interaction, yet their performance drops sharply in professional software environments, even on fundamental operations.

Performance Degrades Sharply with Task Depth. We observe a pronounced degradation in performance as task depth increases. The strongest model, Claude Opus 5, achieves 98.4% success on Atomic tasks with external operational support, indicating near-saturated competence in fundamental operations. However, its success rate drops sharply to 48.3% on Long-Sequence tasks and further

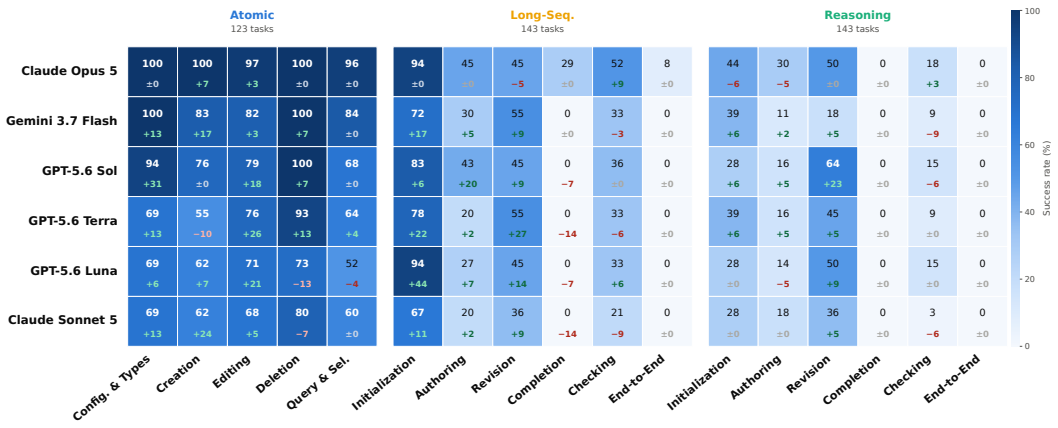


Figure 5: Category-level success rates across BIMArena. Each cell reports SR with operational support. The value below shows the percentage-point change relative to the setting without support.

Table 3: Paired Long-Sequence (L) and Reasoning (R) outcomes pooled across six evaluated models and both support settings. Each pair shares the same target BIM state, with $+$ and $-$ denoting success and failure, respectively.

	Pair Outcome				Conditional Success of R	
Pairs	L^+R^+	L^+R^-	L^-R^+	L^-R^-	$P(R^+ \mid L^+)$	$P(R^+ \mid L^-)$
1716	256	315	65	1080	**44.8%**	**5.7%**

to 26.6% on Reasoning tasks. The same trend is observed across the evaluated models. PCS also declines with task depth but less sharply than SR, suggesting that models often make partial progress on deeper tasks yet increasingly fail to complete them reliably. We further examine the paired Long-Sequence and Reasoning tasks to isolate the effect of domain interpretation. As shown in Table 3, the six evaluated models contribute 1,716 paired cases that share the same target BIM state. When a target cannot be completed from an explicit specification, it is rarely recovered under Reasoning, with a conditional success rate of only 5.7%. Even when the Long-Sequence task succeeds, only 44.8% of the corresponding Reasoning tasks remain successful. These results suggest that domain interpretation introduces a substantial additional barrier beyond the ability to execute the underlying BIM workflow. This depth-induced performance degradation is consistent across both Archicad and Revit, as detailed in the cross-software analysis in Appendix D.3.

Software Proficiency Does Not Translate to BIM Authoring Capability. Figure 5 reveals a clear gap between basic UI interaction and BIM authoring capability. Across all tracks, UI-oriented tasks consistently outperform model-centered tasks. This gap is especially pronounced in the Long-Sequence track. Project Initialization primarily involves interface interaction and reaches success rates of up to 94%. In contrast, categories requiring direct manipulation of building elements and completion of building models perform substantially worse, with success rates more than 40 to 80 percentage points lower. The gap is most severe for Model Completion and End-to-End Modeling, where the long and fully model-centered workflows are rarely completed successfully.

Operational Support Remains Limited for Domain-Specialized Interaction. External operational support consistently helps agents make further progress across all three tracks. As shown in Table 2, the PCS improvement remains relatively stable from Atomic to Long-Sequence and Reasoning tasks. In contrast, the corresponding SR gain decreases substantially from +7.5 points on Atomic tasks to only +0.8 points on Reasoning tasks. The category-level results in Figure 5 further reveal where this limitation arises: operational support mainly benefits UI-centered tasks but can even be detrimental to more difficult model-centered tasks that require extensive interaction with the building model. The largest drops occur in Model Completion, where the success rate falls to

zero for all models except Claude Opus 5. This suggests that operational support can be particularly counterproductive in domains where agents have limited underlying capability.

5.2 Error Analysis

Based on failed trajectories across models, we identify four predominant error patterns in BIM authoring. First, **UI grounding inaccuracies** are a major source of failure for weaker models, particularly when interacting with interface elements that rely on icons or other visually encoded controls without explicit textual annotations. Second, **model geometry understanding** refers to failures in understanding building elements within the model, such as recognizing wall boundaries, determining door orientation, or correctly identifying the intended element. Third, **operation planning** captures failures in composing software operations into the implicit multi-step workflows required by BIM authoring, which is reflected in the substantial performance gap between Atomic and Long-Sequence tasks. Finally, **domain knowledge grounding** describes failures in translating professional domain information into appropriate modeling decisions and is particularly evident from the additional performance degradation from Long-Sequence to Reasoning tasks. Detailed definitions, analyses, and representative failure examples are provided in the Appendix E.

5.3 Ablation Study

To isolate the contribution of the constructed procedural skills, we conduct an ablation study on GPT-5.6 Sol using official software documentation alone, since it shows the largest overall gain from operational support among the evaluated models. As shown in Table 4, this setting performs worse than the no-support setting, indicating that simply providing more information does not necessarily improve agent performance. Official documentation is broad and often contains multiple functions, alternative workflows, and implementation details, which can introduce irrelevant or competing information during retrieval. The results therefore suggest that effective operational support depends more on relevance and low-noise guidance than on the amount of information provided.

Table 4: Ablation results for GPT-5.6 Sol, pooled over Archicad and Revit. *w/o* denotes no external operational support; *w/ doc* provides official software documentation only; *w/ doc+skill* provides both documentation and procedural skills. Superscripts indicate changes relative to *w/o*.

		Atomic			Long-Seq.			Reasoning			Overall
Model	**Support**	SR	PCS	CFR	SR	PCS	CFR	SR	PCS	CFR	SR
GPT-5.6 Sol	w/o	69.9	78.5	91.1	31.5	62.3	66.4	17.5	58.2	58.7	38.1
	w/ doc	$64.2^{-5.7}$	$77.5^{-1.0}$	$89.4^{-1.7}$	$25.2^{-6.3}$	$58.8^{-3.5}$	$64.3^{-2.1}$	$14.7^{-2.8}$	$55.3^{-2.9}$	$53.1^{-5.6}$	$33.3^{-4.8}$
	w/ doc+skill	$\mathbf{80.5}^{+10.6}$	$\mathbf{88.6}^{+10.1}$	$\mathbf{96.7}^{+5.6}$	$\mathbf{39.2}^{+7.7}$	$\mathbf{65.3}^{+3.0}$	$\mathbf{73.4}^{+7.0}$	$\mathbf{21.7}^{+4.2}$	$\mathbf{60.5}^{+2.3}$	$\mathbf{70.6}^{+11.9}$	$\mathbf{45.5}^{+7.4}$

6 Conclusion

We introduce **BIMArena**, a domain-specialized benchmark for evaluating GUI computer-use agents in BIM authoring environments. BIMArena comprises 409 expert-designed tasks organized into three progressive levels that reflect key dimensions of professional software use: complex UI interaction, long-sequence execution, and domain reasoning. Our results reveal several challenges presented by current computer-use agents: General computer-use performance transfers poorly to professional BIM workflows; performance drops sharply with task depth; agents remain stronger at UI interaction than building element manipulation; and operational support offers limited or even negative gains on harder tasks. A key limitation is the substantial evaluation cost, which restricts us to a single run per task. Our scope is also limited to BIM authoring in the architectural domain, while real-world BIM practice involves additional domains and workflows that are not covered in the current benchmark. Even so, our results consistently show that strong computer-use capability does not readily transfer to deeper workflows in professional software, highlighting the need for benchmarks that evaluate agents under realistic domain conditions and support the development of more domain-specialized, effective, and cost-efficient agents. BIMArena takes a step in this direction for BIM authoring, and we believe that such efforts can help foster the development of more specialized agents capable of solving increasingly domain-specialized tasks.

ETHICS STATEMENT

BIMArena is designed for research on computer-use agents in professional BIM authoring environments. The benchmark evaluation does not involve human subjects or the collection of personal or sensitive data. The operational support resource contains expert-reviewed procedural skills for BIM software operations, which are used solely as task-level operational guidance. All evaluations are conducted in controlled software environments using BIM authoring software under valid licenses. The benchmark is intended for capability evaluation rather than for autonomous deployment in safety-critical architectural workflows. Any released benchmark resources will comply with the licensing and redistribution requirements of the underlying software and associated assets.

REPRODUCIBILITY STATEMENT

We release all benchmark components that we are legally permitted to redistribute. These include the complete task specifications and metadata, task specific evaluation constraints and evaluator implementations, environment configuration and setup scripts, agent prompts and action space, operational support, experimental configurations, and initial BIM project files that do not contain or reproduce redistribution-restricted third-party content. The main paper and appendices further document the task-construction process, capability taxonomy, evaluation protocol, software and model versions, and inference settings. An anonymous code repository is available at `https://github.com/bimarenaanon/BIMArena`.

Certain third-party materials, including commercial software binaries, proprietary documentation, technical standards, and copyrighted textbook content, cannot be redistributed directly. We provide official access links in the Appendix where applicable and supply precise provenance and reconstruction information for tasks that depend on such materials. For example, when an architectural drawing originates from a copyrighted textbook, we specify the bibliographic reference, edition, page number, and bounding-box coordinates of the exact region used, allowing users with lawful access to reconstruct the corresponding task input without republishing the protected material. Executing BIMArena requires valid licenses for Autodesk Revit and Graphisoft Archicad. Because the commercial software binaries cannot be redistributed as part of the benchmark, we do not release preconfigured virtual-machine images containing these applications. Instead, we provide the exact software and operating-system versions, environment specifications, configuration files, setup scripts, and official vendor download instructions required to reconstruct the evaluation environment. Users obtain and activate Revit and Archicad independently under their own valid licenses. Eligible users may use vendor-provided educational licenses where permitted under the corresponding licensing terms. BIMArena does not rely on functionality unavailable under the supported educational licenses, and the task definitions and evaluation protocol remain unchanged across the supported licensing configurations.

AI USE STATEMENT

In this work, we used generative AI tools to (i) assist with implementing the benchmark code, including the environment harness, agent framework, and evaluator; (ii) draft initial versions of the operational skills; (iii) polish the wording of expert-written task instructions; and (iv) provide feedback on the experimental design and assist with interpreting the results. All AI-assisted code was reviewed and tested by the authors, and the evaluator was further validated against no-operation and expert-target baselines. The initial skill drafts were revised by domain experts without using any benchmark task instances. All 409 tasks, including their modeling objectives, specifications, domain inputs, BIM models, and evaluation constraints, were designed manually by domain experts. The experts also reviewed every AI-polished task instruction to ensure that it preserved the original specification. All interpretations and conclusions were independently verified by the authors against the experimental results. We did not use generative AI tools to generate synthetic data, develop the capability taxonomy, propose or refine hypotheses, or conduct the qualitative failure analysis. Mathematical claims, proofs, and translation are not applicable to this work. We additionally used generative AI tools for language editing and to help identify related literature; all cited works were independently verified by the authors. We take full responsibility for the final content of this paper, including its text, claims, code, and released artifacts.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Wang. Agent S: An open agentic framework that uses computers like a human. In *International Conference on Learning Representations*, volume 2025, pp. 22924–22946, 2025a.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S2: A compositional generalist-specialist framework for computer use agents. In *Conference on Language Modeling (COLM)*, 2025b.

Jaewoo Ahn, Junseo Kim, Heeseung Yun, Jaehyeon Son, Dongmin Park, Jaewoong Cho, and Gunhee Kim. FlashAdventure: A benchmark for GUI agents solving full story arcs in diverse adventure games. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23354–23384, Suzhou, China, 2025. Association for Computational Linguistics.

Jiaxin Ai, Yukang Feng, Fanrui Zhang, Jianwen Sun, Zizhen Li, Chuanhao Li, Yifan Chang, Wenxiao Wu, Ruoxi Wang, Mingliang Zhai, et al. ProSoftArena: Benchmarking hierarchical capabilities of multi-modal agents in professional software environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 34586–34595, 2026a.

Jiaxin Ai, Tao Hu, Xuemeng Yang, Shu Zou, Hairong Zhang, Daocheng Fu, Yu Yang, Hongbin Zhou, Nianchen Deng, Pinlong Cai, Zhongyuan Wang, Botian Shi, Kaipeng Zhang, and Licheng Wen. ComAct: Reframing professional software manipulation via COM-as-action paradigm, 2026b. URL https://arxiv.org/abs/2606.13239.

Alibaba Cloud. Qwen3.7-Plus. https://qwen.ai/blog?id=qwen3.7-plus, May 2026a. Model documentation.

Alibaba Cloud. Qwen3.8-Max. https://www.alibabacloud.com/help/en/model-studio/qwen3-8-max, August 2026b. Model documentation.

Anthropic. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku. https://www.anthropic.com/news/3-5-models-and-computer-use, October 2024.

Anthropic. Claude Opus 5 System Card. Technical report, Anthropic, July 2026a. URL https://www.anthropic.com/claude-opus-5-system-card.

Anthropic. Claude Sonnet 5 System Card. Technical report, Anthropic, June 2026b. URL https://www.anthropic.com/claude-sonnet-5-system-card.

Anthropic. Teach Claude your way of working using skills. https://academy.claude.com/tutorials/teach-claude-your-way-of-working-using-skills, 2026c. Claude Academy. Accessed: 2026-09-12.

Autodesk. Autodesk Revit 2027. https://www.autodesk.com/products/revit/overview, 2026a. Accessed: 2026-09-12.

Autodesk. Revit API developer's guide. https://help.autodesk.com/view/RVT/2027/ENU/?guid=Revit_API_Revit_API_Developers_Guide_html, 2026b. Autodesk Revit 2027. Accessed: 2026-09-12.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. Windows Agent Arena: Evaluating multi-modal OS agents at scale. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4874–4910. PMLR, 2025.

André Borrmann, Markus König, Christian Koch, and Jakob Beetz (eds.). *Building Information Modeling: Technology Foundations and Industry Practice*. Springer International Publishing, Cham, 2018. ISBN 978-3-319-92861-6. doi: 10.1007/978-3-319-92862-3.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.505. URL https://aclanthology.org/2024.acl-long.505/.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023.

Zihan Deng, Changyu Du, Stavros Nousias, and André Borrmann. BIMgent: Towards autonomous building modeling via computer-use agents. In *ICML 2025 Workshop on Computer-use Agents*, June 2025. doi: 10.48550/arXiv.2506.07217. URL https://openreview.net/forum?id=vQCFwXAl2A.

DIN. DIN 5034-1:2021-08: Daylight in interiors – part 1: Terms and minimum requirements, August 2021. URL https://www.dinmedia.de/en/standard/din-5034-1/337708525.

DIN. DIN 18534-1:2025-10: Waterproofing for indoor applications – part 1: Requirements and principles for design and execution, October 2025. URL https://www.dinmedia.de/en/standard/din-18534-1/391755876.

DIN. DIN 4108-2:2026-05: Thermal insulation and energy saving in buildings – part 2: Thermal insulation – requirements, calculation methods and guidance for design and construction, May 2026. URL https://www.dinmedia.de/en/standard/din-4108-2/401017482.

Changyu Du, Stavros Nousias, and André Borrmann. Towards a copilot in BIM authoring tool using a large language model-based agent for intelligent human-machine interaction. In *Proceedings of the 31st International Workshop on Intelligent Computing in Engineering (EG-ICE 2024)*, pp. 403–412, 2024.

Changyu Du, Zihan Deng, Stavros Nousias, and André Borrmann. Predictive modeling: BIM command recommendation based on large-scale usage logs. *Advanced Engineering Informatics*, 68: 103574, 2025. doi: 10.1016/j.aei.2025.103574.

ENZYME-APD. Tapir Archicad automation. https://github.com/ENZYME-APD/tapir-archicad-automation, 2026. Version 1.5.3. Accessed: 2026-09-12.

Honghao Fu, Junlong Ren, Qi Chai, Deheng Ye, Yujun Cai, and Hao Wang. VistaWise: Building cost-effective agent with cross-modal knowledge graph for Minecraft. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21884–21898. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1111.

Google DeepMind. Gemini 3.7 Flash Model Card. Technical report, Google DeepMind, August 2026. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/.

Graphisoft. Archicad 29. https://www.graphisoft.com/en-us/plans-and-products/archicad/, 2025a. Accessed: 2026-09-12.

Graphisoft. Archicad JSON interface. https://archicadapi.graphisoft.com/JSONInterfaceDocumentation/, 2025b. Version 29.0.0.3000. Accessed: 2026-09-12.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shenzhi Wang, Xinchen Xu, Shuofei Qiao, Zhaokai Wang, Kun Kuang, Tieyong Zeng, Liang Wang, Jiwei Li, Yuchen Eleanor Jiang, Wangchunshu Zhou, Guoyin Wang, Keting Yin, Zhou Zhao, Hongxia Yang, Fan Wu, Shengyu Zhang, and Fei Wu. OS agents: A survey on MLLM-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7436–7465, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.369. URL https://aclanthology.org/2025.acl-long.369/.

Hongrui Jia, Jitong Liao, Xi Zhang, Haiyang Xu, Tianbao Xie, Chaoya Jiang, Ming Yan, Si Liu, Wei Ye, and Fei Huang. OSWorld-MCP: Benchmarking MCP tool invocation in computer-use agents. In *International Conference on Learning Representations*, 2026.

Yongqing Jiang, Jianze Wang, Zhiqi Shen, Zhenghong Lin, Jiayuan Wang, Yijian Yang, Kaoshan Dai, and Haoran Luo. Rethinking scientific modeling: Toward physically consistent and simulation-executable programmatic generation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 11189–11200, 2026. doi: 10.1145/3770855.3818987.

Ahti Kalervo, Juha Ylioinas, Markus Häikiö, Antti Karhu, and Juho Kannala. CubiCasa5K: A dataset and an improved multi-task model for floorplan image analysis. In *Image Analysis – 21st Scandinavian Conference, SCIA 2019*, volume 11482 of *Lecture Notes in Computer Science*, pp. 28–40. Springer, 2019. doi: 10.1007/978-3-030-20205-7_3.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. OmniACT: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXVIII*, pp. 161–178, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73112-9. doi: 10.1007/978-3-031-73113-6_10. URL https://doi.org/10.1007/978-3-031-73113-6_10.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 881–905, 2024. doi: 10.18653/v1/2024.acl-long.50.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Chunyi Li, Longfei Li, Zicheng Zhang, Xiaohong Liu, Min Tang, Weisi Lin, and Guangtao Zhai. Using GUI agent for electronic design automation, 2025a.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. ScreenSpot-Pro: GUI grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 8778–8786, 2025b. doi: 10.1145/3746027.3755688.

Zhaoyang Liu, Jingjing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. ScaleCUA: Scaling open-source computer use agents with cross-platform data. In *International Conference on Learning Representations*, volume 2026, pp. 62707–62766, 2026.

King Yiu Brandon Man, Ghadi Nehme, Md Ferdous Alam, and Faez Ahmed. VideoCAD: A dataset and model for learning long-horizon 3D CAD UI interactions from video. In *Advances in Neural Information Processing Systems*, volume 38, pp. 40705–40747, 2025. doi: 10.52202/085713-1217.

Meta AI. Introducing Muse Spark 1.1. https://ai.meta.com/blog/introducing-muse-spark-meta-model-api/, July 2026. Official model release.

Microsoft. Skill recorder. https://github.com/microsoft/skill-recorder, 2026. Open-source software. Accessed: 2026-09-12.

Ernst Neufert and Peter Neufert. *Architects' Data*. Wiley-Blackwell, 4th edition, 2012. ISBN 9781405192538.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie,

Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Jihyung Kil, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. GUI agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22522–22538. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1158.

Bharathi Kannan Nithyanantham, Clemens Kujat, Tobias Sesterhenn, Stefan Telgmann, Ashwin Nedungadi, Jörn Plönnigs, Christian Bartelt, and Stefan Lüdtke. BIM-Edit: Benchmarking large language models for IFC-based building information modeling. *arXiv preprint arXiv:2606.20146*, 2026.

OpenAI. Computer-using agent. https://openai.com/index/computer-using-agent/, January 2025. Accessed: 2026-08-28.

OpenAI. GPT-5.6 Luna. https://openai.com/index/gpt-5-6/, July 2026a. GPT-5.6 cost-efficient model.

OpenAI. GPT-5.6 Sol. https://openai.com/index/gpt-5-6/, July 2026b. GPT-5.6 flagship model.

OpenAI. GPT-5.6 Terra. https://openai.com/index/gpt-5-6/, July 2026c. GPT-5.6 balanced model.

Yesol Park, Jakub Krukar, Martin Brösamle, John Gero, and Christoph Hölscher. Interpreting architectural drawings: The role of gaze and gestures in cognitive offloading. *Applied Cognitive Psychology*, 40(1):e70127, 2026. doi: 10.1002/acp.70127.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering automated gui interaction with native agents, 2025. URL https://arxiv.org/abs/2501.12326.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, 2025.

Alcinia Zita Sampaio, Paulo Sequeira, Augusto M. Gomes, and Alberto Sanchez-Lite. BIM methodology in structural design: A practical case of collaboration, coordination, and integration. *Buildings*, 13(1):31, 2023. doi: 10.3390/buildings13010031.

Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, Jianing Wang, Qintong Li, Xiangru Tang, Tianbao Xie, Xiachong Feng, Xiang Li, Ben Kao, Wenhai Wang, Biqing Qi, Lingpeng Kong, and Zhiyong Wu. ScienceBoard: Evaluating multimodal autonomous agents in realistic scientific workflows. In *International Conference on Learning Representations*, 2026.

Zeyi Sun, Yuhang Cao, Jianze Liang, Qiushi Sun, Ziyu Liu, Zhixiong Zhang, Yuhang Zang, Xiaoyi Dong, Kai Chen, Dahua Lin, and Jiaqi Wang. CODA: Coordinating the cerebrum and cerebellum for a dual-brain computer use agent with decoupled reinforcement learning, 2025. URL https://arxiv.org/abs/2508.20096.

Weihao Tan, Xiangyang Li, Yunhao Fang, Heyuan Yao, Shi Yan, Hao Luo, Tenglong Ao, Huihui Li, Hongbin Ren, Bairen Yi, Yujia Qin, Bo An, Libin Liu, and Guang Shi. Lumine: An open recipe for building generalist agents in 3D open worlds, 2025a.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle:

Empowering foundation agents towards general computer control. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 58658–58725. PMLR, 2025b.

Wenkai Wang, Tao Xiong, Jingchen Ni, Yunpeng Bao, Xiyun Li, Tianqi Liu, Hongcan Guo, Zilong Huang, and Shengyu Zhang. DeskCraft: Benchmarking desktop agents on professional workflows and human-in-the-loop collaboration. *arXiv preprint arXiv:2606.03103*, 2026a.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50208–50232. PMLR, 2024.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Y. Charles, Zhilin Yang, and Tao Yu. OpenCUA: Open foundations for computer-use agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4669.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. MCP-Bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers. In *International Conference on Learning Representations*, 2026b.

Jinbiao Wei, Yilun Zhao, Kangqi Ni, and Arman Cohan. Anchor: Branch-point data generation for GUI agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17031–17047, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.774. URL https://aclanthology.org/2026.acl-long.774/.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. OS-Copilot: Towards generalist computer agents with self-improvement, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024. doi: 10.52202/079017-1650.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. In *Advances in Neural Information Processing Systems*, volume 38, pp. 10291–10321, 2025a. doi: 10.52202/085713-0315.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous GUI interaction. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69772–69805. PMLR, 2025b. URL https://proceedings.mlr.press/v267/xu25ae.html.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, et al. EvoCUA: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv:2601.15876*, 2026.

Yuhao Yang, Tianyu Fan, and Chao Huang. CLI-Anything: Towards agent-native computer use, 2026. URL https://arxiv.org/abs/2606.03854.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, Qi Zhen, Xinyuan Wang, Jiaqi Deng, Yuhao Yang, Cheng Chen, Boyuan Zheng, Alex Su, Xiao Yu, Hao Zou, Saaket Agashe, Xing Han Lu, Manpreet Kaur, Zhengyang Qi, Vincent Sunn Chen, Frederic Sala, Dayiheng Liu, Junyang Lin, Zhou Yu, Yu Su, Siva Reddy, Xin Eric Wang, Peng Qi, Tianbao Xie, and Tao Yu. OSWorld 2.0: Benchmarking computer use agents on long-horizon real-world tasks. *arXiv preprint arXiv:2606.29537*, 2026.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Large language model-brained GUI agents: A survey. *Transactions on Machine Learning Research*, 2025.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

Xiao Zhou, Siyue Zhang, Yilun Zhao, Jinbiao Wei, Tingyu Song, Arman Cohan, and Chen Zhao. GUI vs. CLI: Execution bottlenecks in screen-only and skill-mediated computer-use agents. *arXiv preprint arXiv:2606.24551*, 2026.

A Details of the BIMArena Tasks

A.1 Functional Coverage of BIM Authoring Tasks

Real-world BIM authoring workflows span a broad range of activities, from architectural model development to more specialized tasks such as reinforcement detailing and Mechanical, Electrical, and Plumbing (MEP) system modeling (Sampaio et al., 2023). The current BIMArena task set primarily focuses on architectural model development, namely the construction and configuration of the fundamental BIM model.

At the element level, the benchmark covers major architectural elements, including walls, openings, slabs, rooms and stairs, which must be created with specified geometry, semantic information, and detailed modeling requirements. Beyond element creation, tasks also require substantially richer parameter configuration, such as selecting wall assemblies with predefined layer structures, assigning materials and finishes, specifying dimensions and offsets, and defining geometric or relational constraints between building elements.

In addition to model construction, BIMArena covers essential project-level configuration operations, including setting story elevations, switching views, inspecting model properties, and managing other project-level settings. Together, these tasks cover a broad range of functionalities involved in architectural BIM authoring, while more specialized workflows such as reinforcement detailing and MEP system design remain outside the current scope.

A.2 Reference Resources

BIMArena draws on both software-specific learning resources and professional domain references during benchmark construction. Table 5 summarizes the main resources used.

Software Learning Resources. For software, we primarily refer to the official documentation, help pages, and learning materials. These resources are used to verify software-specific terminology, available functionalities, and standard operating procedures when constructing and validating benchmark tasks.

Domain Reference Resources. To construct realistic Long-Sequence and Reasoning tasks, we further refer to established architectural references and technical standards. In particular, *Architects' Data* by Neufert (Neufert & Neufert, 2012) and the floorplan dataset CubiCasa5K (Kalervo et al., 2019) serve as general architectural references. We also consult relevant DIN standards, including DIN 4108-2 (DIN, 2026) for thermal protection, DIN 18534-1 (DIN, 2025) for waterproofing in indoor applications, and DIN 5034-1 (DIN, 2021) for daylighting requirements. These references are selectively incorporated according to the requirements of individual tasks rather than being uniformly applied across the benchmark.

Table 5: Software and professional reference resources used in the construction of BIMArena.

Resource	Provider / Publisher	Version / Edition	Type	Official Resource
Revit	Autodesk	2027	BIM Software	Revit 2027 Help / Video Gallery
Archicad	Graphisoft	29	BIM Software	Archicad 29 Help / Learning Center
Architects' Data	Ernst Neufert / Peter Neufert	4th Edition (2012)	Reference Book	Architects' Data
CubiCasa5K	CubiCasa	2019	Floorplan Dataset	CubiCasa5K Dataset
DIN 4108-2	DIN	2026-05	Thermal Protection	DIN 4108-2
DIN 18534-1	DIN	2025-10	Indoor Waterproofing	DIN 18534-1
DIN 5034-1	DIN	2021-08	Daylighting	DIN 5034-1

A.3 Fundamental Building Modeling Capabilities

The fundamental capability taxonomy is derived from the software and domain resources summarized in Table 5. We identify frequently recurring and practically important building-modeling operations across official software documentation, learning materials, architectural references, technical standards, and professional practices, and consolidate them into software-independent capability

definitions. The scope is introduced in A.1. Table 6 presents the complete set of 23 fundamental building modeling capabilities used in BIMArena.

Table 6: The 23 fundamental building modeling capabilities.

FUNDAMENTAL CAPABILITY	DESCRIPTION
Config. & Types	
Create Stories	Create and configure building stories, including their names, elevations, and vertical relationships.
Switch Active Story	Navigate between building stories and activate the target story for subsequent modeling operations.
Create Wall Type	Define a reusable wall type with specified construction, material, thickness, and layer properties.
Create Slab Type	Define a reusable slab type with specified construction, material, thickness, and layer properties.
Create Door/Window Type	Define reusable door or window types with specified dimensions and properties.
Load Library Element Type	Load predefined building element types from the software library into the current project for reuse.
Creation	
Create Wall	Create wall elements according to specified geometry, wall types, dimensions, and placement conditions.
Create Slab	Create slab elements according to specified boundaries, slab types, elevations, and properties.
Create Slab Opening	Create openings within existing slab elements according to specified boundaries and dimensions.
Create Door	Create door elements in host walls with the required type, position, orientation, and dimensions.
Create Window	Create window elements in host walls with the required type, position, orientation, and dimensions.
Create Stair	Create stair elements connecting specified building levels while satisfying requirements.
Create Room/Zone	Define room or zone regions within the building model and assign the specified names and numbers.
Editing	
Move Element	Move existing building elements from their current locations to specified target positions.
Move with Constraints	Move existing elements while preserving specified geometric or relational constraints.
Flip Element	Change the orientation or facing direction of an existing building element.
Replace Element Type	Replace the type of an existing element with another available type while preserving its placement.
Deletion	
Delete Element	Remove specified building elements without unintentionally modifying unrelated model components.
Query & Selection	
Inspect Project	Inspect the current project settings, model element types, and element properties.
Check Clash	Identify unintended geometric intersections or spatial conflicts between building elements.
Query Materials	Inspect and identify predefined materials available in the software for use in building element type composition.
Query Types	Inspect the building element types currently loaded and available within the project.
Query Library	Inspect reusable and predefined building element types available in the software library.

A.4 METADATA ANNOTATION

During expert task construction, each task in BIMArena is accompanied by structured metadata and instantiated within a standardized software environment with a fixed application version. The metadata profile comprises the following fields:

Task ID. A unique identifier for each task.

Category. The functional or workflow category of the task.

Software. The target BIM authoring application used for execution.

Instruction. The natural-language task instruction provided to the agent.

Additional Context. Image-based inputs for reasoning tasks, such as architectural drawings.

Initial File. The initial BIM model used to initialize the task environment.

Task-specific evaluation constraints. The target conditions used for programmatic task evaluation.

Required Capabilities and Occurrence Counts. The fundamental capabilities required to complete the task and the expected number of occurrences of each capability.

Table 7 provides an example of the metadata annotation.

Table 7: Example metadata annotation for a BIMArena task.

Metadata Field	Example
Task ID	`C_model_Revision_7`
Category	Model Revision
Software	Autodesk Revit 2027
Instruction	The floorplan shown in the attached drawing has already been built in the current model. The drawing carries redline annotations marking the required design changes.
Additional Context	`drawing.png`
Initial File Path	`tasks/revit/C_7/init.rvt`
Task-specific evaluation constraints	Structured element-level constraints; see Listing 1.
Required Capabilities and Occurrence Counts	Delete Element ($\times 2$); Create Door ($\times 1$); Create Window ($\times 1$); Create Wall ($\times 1$)

Listing 1: Example of the structured expected-result annotation used for programmatic evaluation.

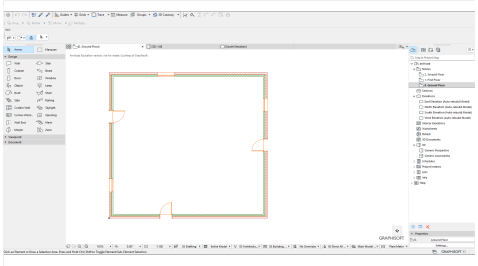
```
"expected_result": {

  "doors": [
    {
      "label": "added entrance door (west wall)",
      "center": [0, 1450],
      "width_mm": 900,
      "height_mm": 2100,
      "open_into": [-1000, 1450],
      "hinge_near": [0, 1900]
    }
  ],

  "windows": [
    {
      "label": "added window (west wall)",
      "center": [0, 7550],
      "width_mm": 1500,
      "height_mm": 1500,
      "sill_mm": 1000
    }
  ],
```

```
  "opening_pos_tolerance_mm": 100,

  "exact_counts": {
    "walls": 15,
    "doors": 5,
    "windows": 4,
    "slabs": 1,
    "rooms": 0,
    "stairs": 0,
    "objects": 0
  },

  "preserve": [
    {
      "bucket": "walls",
      "coordinates": [[5551, 3476]]
    },
    {
      "bucket": "doors",
      "coordinates": [[10766, 0]]
    },
    "windows",
    "slabs"
  ],

  "composites": [],
  "no_extra_composites": true
}
}
```

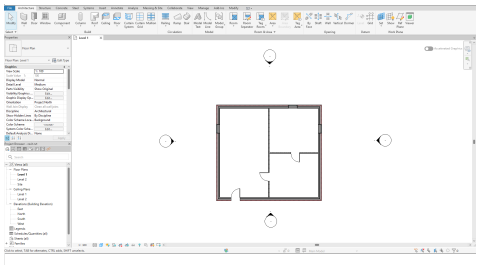

A.5 Task Examples

We further provide representative task examples from the Atomic, Long-Sequence, and Reasoning tracks in BIMArena. For each example, we show the task instruction together with the initial BIM model and, when applicable, additional domain-specialized context such as architectural drawings.

Atomic Tasks. Atomic tasks isolate individual BIM authoring capabilities and typically require relatively short interaction sequences. Figure 6 presents two representative examples.



Instruction. Create one zone named `Storage` with number `10` filling the area enclosed by the exterior walls. Create nothing else.



Instruction. Create one floor covering the building's full footprint, extending to the outer faces of the four exterior walls, using the existing floor type `Insitu Concrete 225mm`. Create no other elements.

Figure 6: Representative Atomic tasks in BIMArena. Each task isolates an individual BIM authoring capability with a relatively short interaction sequence.

Long-Sequence Tasks. Long-Sequence tasks require agents to execute substantially longer BIM authoring workflows, while the intended modeling objective and relevant geometric specifications are explicitly provided in the task instruction.

Example 1: Complete Single-story Building.

This task requires constructing a complete single-story building, including exterior and interior walls, doors, windows, and a floor slab. The required coordinates, dimensions, and wall construction types are explicitly specified. The initial BIM model is an empty project, as shown in Figure 7.

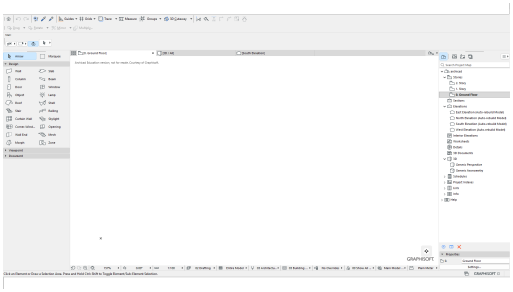


Figure 7: Initial BIM model Example 1.

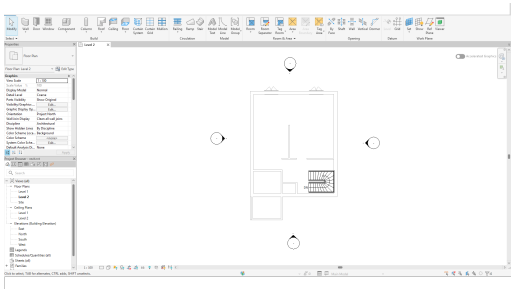


Figure 8: Initial BIM model for Example 2.

Instruction.

Create the complete single-story building: all walls, doors, windows, and the floor slab. Do not create composites—use existing ones. Ignore furniture.

Exterior walls using an EXISTING composite whose name contains 'Double 50':

- $(0, 1100) \rightarrow (0, 8260)$
- $(0, 8260) \rightarrow (6150, 8260)$
- $(0, 1100) \rightarrow (2650, 1100)$
- $(2650, 1100) \rightarrow (2650, 0)$
- $(2650, 0) \rightarrow (6150, 0)$

Exterior walls using an EXISTING composite whose name contains 'Insulated':

- $(6150, 0) \rightarrow (10550, 0)$
- $(10550, 0) \rightarrow (10550, 8260)$
- $(6150, 8260) \rightarrow (6150, 10260)$
- $(6150, 10260) \rightarrow (10350, 10260)$
- $(10350, 10260) \rightarrow (10350, 8260)$
- $(10350, 8260) \rightarrow (10550, 8260)$

Interior walls (any suitable existing interior wall type):

- $(6425, 7986) \rightarrow (10075, 7986)$
- $(0, 3800) \rightarrow (6150, 3800)$
- $(6150, 8260) \rightarrow (6150, 7260)$
- $(2924, 1374) \rightarrow (2924, 3800)$
- $(3274, 3800) \rightarrow (3274, 6300)$
- $(6150, 4800) \rightarrow (6150, 0)$
- $(10550, 2600) \rightarrow (6150, 2600)$

Doors. 900 $\times$ 2100 mm unless noted, centred at: $(7375, 7986)$; $(10350, 9160)$; $(8625, 2600)$, 700 mm wide; $(6150, 1574)$, 800 mm wide; $(4575, 3800)$; $(1324, 3800)$; and $(1250, 1100)$.

Windows. 900 $\times$ 1200 mm with 1000 mm sill unless noted, centred at: $(10550, 6810)$; $(10550, 5410)$, 900 $\times$ 1500 mm; $(4647, 0)$, 900 $\times$ 1500 mm; $(6900, 0)$, 500 $\times$ 1500 mm; $(6150, 9260)$; $(7298, 10260)$; $(8461, 10260)$; $(9571, 10260)$; $(0, 6510)$; $(1868, 8260)$; and $(4389, 8260)$.

Floor slab. Create a floor slab at level 0 with outline $(0, 8260)$, $(6150, 8260)$, $(6150, 10260)$, $(10350, 10260)$, $(10350, 8260)$, $(10550, 8260)$, $(10550, 0)$, $(2650, 0)$, $(2650, 1100)$, and $(0, 1100)$.

Example 2: Upper-Floor Completion.

This task requires constructing an upper floor on top of an existing ground-floor model while preserving the existing model. The required wall types, coordinates, openings, and floor geometry are explicitly provided. The initial BIM model is shown in Figure 8.

Instruction.

The ground floor is already created in the project. Create the upper floor on the level at 4000 mm exactly as follows, reusing the types already used on the ground floor. Do not create or modify any types, and do not alter the ground floor.

Exterior walls, 'Exterior - Brick on Mtl. Stud' (draw each exactly on the line given):

- $(171, 2343) \rightarrow (9171, 2343)$
- $(9346, 2518) \rightarrow (9346, 13518)$
- $(9171, 13693) \rightarrow (171, 13693)$
- $(-4, 13518) \rightarrow (-4, 2518)$

Interior walls, 'Interior - 138mm Partition (1-hr)':

- $(9171, 6362) \rightarrow (171, 6362)$
- $(4796, 6431) \rightarrow (4796, 13331)$
- $(4102, 2718) \rightarrow (4102, 4493)$
- $(4171, 4562) \rightarrow (2971, 4562)$
- $(3012, 4437) \rightarrow (1974, 3662)$ (a short diagonal)
- $(1864, 3718) \rightarrow (1864, 2518)$
- $(2902, 4493) \rightarrow (2902, 6431)$

Doors. Both doors are 915×2134 mm, centred at $(4100, 6431)$ opening toward $+y$, and $(5654, 6431)$ opening toward $+y$.

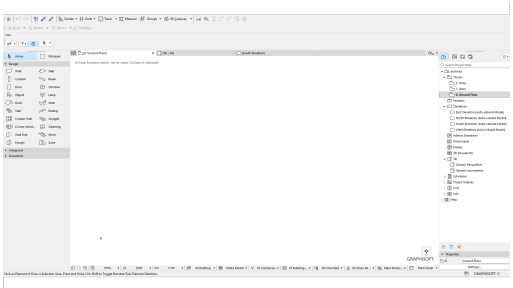
Windows. 1400×1800 mm with 900 mm sill unless noted, centred at $(2246, 13518)$; $(7096, 13518)$; $(5202, 2518)$; and $(2627, 2518)$, with the final window sized 850×900 mm.

Floor slab. Create the slab on the upper level using type `Concrete-Commercial 362mm`, with outline $(9346, 13693)$, $(9346, 2343)$, $(-4, 2343)$, and $(-4, 13693)$. Keep the slab clear of the existing stair by cutting the stairwell opening so that the stair passes through.

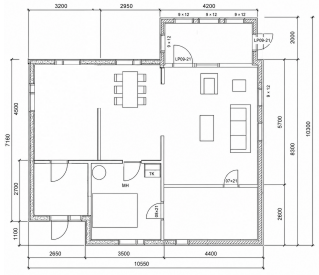
Reasoning Tasks. Reasoning tasks share same underlying modeling objectives with their paired Long-Sequence tasks, but the explicit modeling specifications are replaced by domain-specialized inputs. The agent must first interpret these inputs to infer the intended BIM state before executing the required software operations.

Example 1: Building Construction from a Dimensioned Floorplan.

In this example, the agent must infer wall geometry, construction types, and opening placement directly from the architectural drawing rather than receiving explicit element coordinates and types, as shown in Figure 9.



(a) Initial BIM model



(b) Architectural drawing

Figure 9: Representative Reasoning task requiring construction of a complete building from a dimensioned architectural floorplan. The agent must infer geometry, wall construction types, and opening placement from the drawing before executing the corresponding BIM operations.

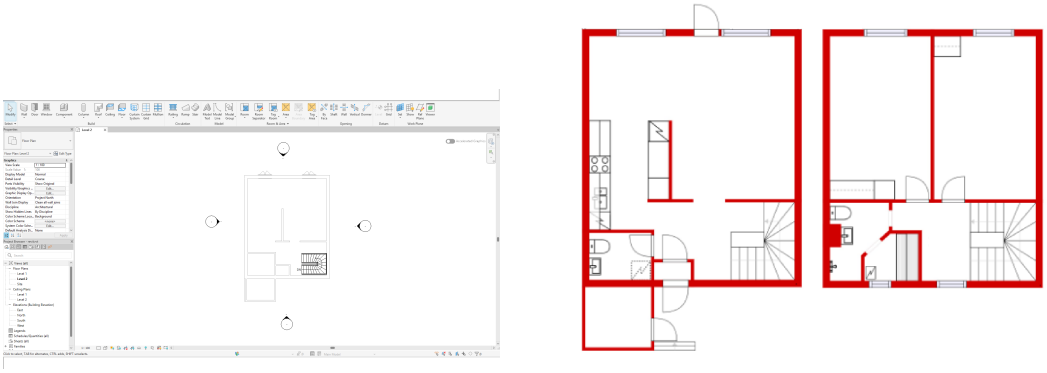
Instruction.

Based on the provided dimensioned floor plan, create the complete single-story building: all walls, doors, windows, and the floor slab. Judge each wall's construction type by how it is represented in the drawing (hatch/fill pattern of its section) and create every wall with

the matching wall type. Place doors and windows at the positions and sizes indicated in the drawing. Ignore all furniture.

Example 2: Completion of a Partially Modeled Apartment.

In this example, the initial BIM model already contains part of the apartment. The agent must jointly interpret the existing model and the architectural drawing to determine which elements are missing and how they should be integrated with the existing construction, as shown in Figure 10.



(a) Initial BIM model (b) Architectural drawing

Figure 10: Representative Reasoning task requiring completion of a partially modeled apartment. The agent must identify missing elements from the architectural drawing while preserving and matching the existing BIM model.

Instruction.

The apartment shown in the floor plan is only partly modeled in the project. Complete it: add every wall, door, window, and floor slab that is still missing, matching the layout, positions, and connections shown in the drawing. Match what is already built—reuse the wall, door, window, and slab types already present in the project. Do not create or modify any types, and do not change or delete what is already correctly built. Door and window sizes are not critical; the default size of the matching family is sufficient.

A.6 Comparison with Existing Benchmarks

Table 8: Comparison of representative computer-use and professional-software benchmarks. "Avg. steps" reports the execution horizon provided by the corresponding work; step definitions are not fully standardized across benchmarks. N/R indicates that an overall average is not reported.

Benchmark	Domain	Apps	Tasks	Avg. steps	Interface	Evaluation	Paired reasoning	Collateral metric
OSWorld (Xie et al., 2024)	General desktop	9	369	∼30 (agent steps)	GUI	State-based	×	×
OSWorld 2.0 (Yuan et al., 2026)	General/professional	N/R	108	>250 (agent steps)	GUI/tools	State-based	×	×
WindowsAgentArena (Bonatti et al., 2025)	General Windows	11	154	N/R	GUI	State-based	×	×
ScienceBoard (Sun et al., 2026)	Scientific software	6	169	9.0	GUI/CLI	State-based	×	×
ProSoftArena (Ai et al., 2026a)	Professional software	13	456	12.9	GUI	State + human	×	×
DeskCraft (Wang et al., 2026a)	Professional software	11	538	N/R	GUI	State-based	×	×
VideoCAD (Man et al., 2025)	CAD	1	41,005 seq.	186	GUI	Trajectory-based	×	×
ComCADBench (Ai et al., 2026b)	CAD	3	1,000	N/R	COM	Artifact-based	×	×
BIM-Edit (Nithyanantham et al., 2026)	BIM (IFC)	N/A	324	N/R	Code	State-based	×	×
BIMArena	**BIM**	**2**	**409**	**50.9 (max 203)**	**GUI**	**State-based, native API**	✓	**✓ (CFR)**

Table 8 compares BIMArena with representative general-purpose, professional-software, CAD, and BIM benchmarks. Compared with broad computer-use benchmarks, BIMArena focuses on deeper coverage of a single professional domain and includes longer workflows. Compared with existing CAD/BIM benchmarks, it combines large-scale GUI-based evaluation across two BIM authoring platforms with paired Long-Sequence and Reasoning tasks. BIMArena additionally evaluates the resulting model state through native APIs and measures unintended modifications using CFR, enabling evaluation of both task completion and execution reliability.

B External Operational Support

As described in Section 4.2, we provide two complementary forms of external operational support: official software documentation and expert-curated operational skills.

Official Documentation. We collect the official documentation resources for Revit and Archicad listed in Table 5. The original HTML documentation is extracted and converted into Markdown files, resulting in 478 documents for Revit and 237 documents for Archicad. Agents are given access to a dedicated documentation retrieval tool. Retrieval is based on keyword matching, following the conventional way users search software documentation using function names and operation-related terms. For each query, the three highest-ranked documents are returned to provide sufficient coverage while limiting retrieval context.

Operational Skills. In addition to official documentation, we construct software-specific operational skills for each fundamental capability listed in Table 6. Inspired by reusable agent Skills (Anthropic, 2026c), an initial operation recipe is first generated for each capability and software platform. We found that these automatically generated drafts often lacked sufficient software-specific precision and procedural reliability. Domain experts therefore review and refine each draft by correcting interaction steps, adding application-specific setup instructions, recovery strategies, and final verification criteria. Importantly, the construction, refinement, and debugging of these skills are performed independently of the BIMArena benchmark tasks: no benchmark task instances are used to develop, tune, or validate the skill content. This separation ensures that the skills encode reusable capability-level operational knowledge rather than task-specific solutions. This expert-in-the-loop process results in one validated skill for Revit and one for Archicad for each capability, yielding 46 skills in total. Each skill contains a short capability description, software-specific setup instructions, step-by-step interaction procedures, recovery strategies, and final verification criteria. Agents access these skills on demand through a separate retrieval tool based on retrieval-augmented generation (RAG) (Lewis et al., 2020). Skills are not automatically injected into the agent context. When the agent issues a retrieval query, the module matches the query against the skill descriptions and returns the most relevant complete skill file. Examples of the resulting skills for the same `create_wall` capability in Archicad and Revit are shown in Listings 2 and 3.

Listing 2: Example operational skill for `create_wall` in Archicad.

```
---
name: create_wall
description: Create straight Archicad walls from an existing
          basic or composite structure.
software: Archicad
---

# Create a wall

Create straight Archicad walls using the required existing
basic or composite structure. Work in the Floor Plan and use
exact geometric snaps and numerical dimensions.

## Setup

1. Activate the Wall tool.

2. Ensure that the Straight Wall Geometry Method is active.
   If another geometry method is selected, press and hold the
   left-most geometry-method icon in the Info Box and select
   Straight Wall from the available options.

3. Select the required Wall Composite / Basic Structure.
   Verify the exact structure name in the Info Box before
   placing any wall.

4. Set the Reference Line according to the wall function:
   - Exterior walls: Outside
   - Interior walls: Center

5. Verify and set the required wall height and vertical
   properties in the Info Box.

## Common Warnings
```

```
- Do not use Chained Wall mode when individual wall segments
  are required.


## Wall Creation

For each wall:

1. Click the exact snapped start point.
2. Move the cursor in the required direction.
3. Press Tab to activate numerical input.
4. Type the required wall length.
5. Press Enter to commit the wall.

## Common Warnings

Do not click an additional endpoint after pressing Enter;
the wall has already been created.

For connected exterior walls, continue from the previous
wall endpoint and construct the perimeter counter-clockwise.

## Connectivity

- Use exact geometry snaps for all wall connections.
- Each following exterior wall should start from the previous
  wall endpoint.
- T-junctions must terminate directly on the host wall body.
- The final exterior wall must snap exactly to the original
  starting point to close the perimeter.
- Do not place endpoints approximately near existing geometry.

If a snap point is unclear, zoom in before committing the wall.

## Recovery

If the wall command is interrupted:

1. Reactivate the Wall tool.
2. Restore the required structure and Reference Line.
3. Snap to the last valid wall endpoint.
4. Continue using the required direction and typed length.

If an incorrect structure or reference line was active,
correct the setting before creating additional walls rather
than continuing with the wrong configuration.

## Verification

Before finishing, verify:

- the correct Wall Composite / Basic Structure;
- the correct Reference Line;
- the required wall length and orientation;
- exact wall-to-wall connectivity;
- closure of the exterior perimeter;
- absence of unintended additional walls.

Press Esc when all required walls have been completed.
```

Listing 3: Example operational skill for `create_wall` in Revit.

```
---
name: create_wall
description: Create straight Revit walls from an existing
          Wall Type.
software: Revit
---

# Create a wall

Create straight Revit architectural walls using the required
existing Wall Type. Open the target floor plan before placement
and enter required dimensions numerically.

## Setup

1. Activate Architecture > Wall.

2. Verify the required Draw mode:
```

```
   - Line: individually dimensioned walls
   - Rectangle: simple rectangular footprints

   If another mode is active, select the required mode before
   starting wall creation.

3. Select the required Wall Type using the Type Selector.
   Verify the exact type before placing any wall.

4. Set the Location Line according to the wall function:
   - Exterior walls: Finish Face: Exterior
   - Interior walls: Wall Centerline

5. Verify the required vertical constraints:
   - Base Constraint
   - Top Constraint
   - Unconnected Height, when applicable

## Common Warnings

- Enable Chain only when creating a continuous connected wall run.
- Do not leave Chain enabled when individually specified wall
  segments are required.


## Wall Creation

For Line mode:

1. Click the exact required start point.
2. Move the cursor in the required direction.
3. Type the required wall length.
4. Press Enter.


With Chain enabled, continue directly from the endpoint of the
previous wall without clicking a new start point.

For a rectangular footprint, Rectangle mode may be used by
selecting two diagonal corners and subsequently correcting the
required dimensions.

## Common Warnings
Do not estimate required production dimensions from the
screenshot. Use the values typed during wall placement.

Temporary dimensions may measure between wall faces and can
therefore differ from the typed wall length around joined walls.

## Connectivity

- Snap directly to existing wall geometry.
- Continue connected exterior walls from the previous endpoint.
- Interior walls terminating at another wall must snap directly
  onto the host wall.
- The final exterior wall must return exactly to the original
  starting point.

## Recovery

If the wall chain is interrupted by Esc, a dialog, or a mode
change:

1. Reactivate the Wall tool if necessary.
2. Verify the Wall Type and Location Line.
3. Re-enable Chain when required.
4. Snap precisely to the exterior finish-face endpoint of the
   previously created wall.
5. Continue with the required typed length.

If a wall is created with the wrong type or location line,
correct or recreate the affected wall before proceeding.

## Verification

Before finishing, verify:

- the required Wall Type;
- the correct Location Line;
- the typed wall length;
- Base and Top Constraints;
```

```
- wall orientation and connectivity;
- closure of the exterior perimeter;
- absence of unintended additional walls.

Judge wall lengths from the entered numerical values rather
than visual estimation from the screenshot.
```

C Details of BIMArena Environment

C.1 Software

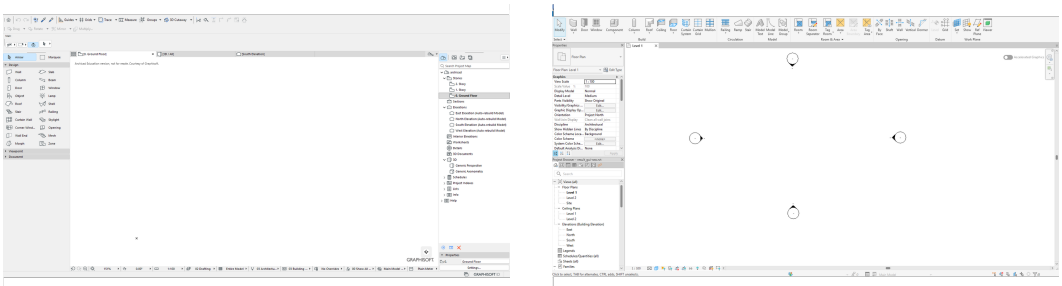
To evaluate domain-deep computer-use capabilities in professional building modeling environments, BIMArena adopts two widely used BIM authoring platforms, Autodesk Revit and Graphisoft Archicad. We select these platforms for three main reasons. First, both support comprehensive BIM authoring workflows, including project setup, parametric element creation, type and attribute management, model editing, and inspection. Although they expose largely overlapping high-level capabilities, they differ substantially in interface organization, terminology, and software-specific interaction workflows, providing complementary environments for evaluating cross-software computer-use capabilities.

Second, both platforms provide extensive official documentation and learning resources, including user guides, step-by-step tutorials, reference materials, and a large collection of instructional videos. These resources provide reliable software-specific references for task construction and for the external operational support setting.

Third, both platforms provide comprehensive native APIs that allow programmatic access to the current BIM model state, including element inventories, geometric properties, semantic attributes, and inter-element relationships. This enables BIMArena to inspect the final model state directly and perform deterministic programmatic evaluation without relying on an LLM-based judge.

Graphisoft Archicad. Graphisoft Archicad (Graphisoft, 2025a) is a BIM authoring platform developed by Graphisoft for architectural design, building modeling, and documentation. It represents buildings through object-based BIM elements such as walls, slabs, doors, windows, stairs, zones, and library components, while supporting project-wide management of stories, attributes, materials, and reusable objects. Its workflow integrates plan-based and 3D editing with dedicated tool settings, navigator-based project organization, and library-driven object placement. The user interface is shown in Figure 11a.

Autodesk Revit. Autodesk Revit (Autodesk, 2026a) is a BIM authoring platform developed by Autodesk for architectural, structural, and building engineering workflows. It organizes building models around parametric elements such as walls, floors, doors, windows, stairs, and rooms, while supporting project-level management of levels, views, families, materials, and model properties. Its interface combines ribbon-based commands, project navigation, property panels, contextual editing, and geometry-dependent interactions. The user interface is shown in Figure 11b.



(a) Graphisoft Archicad 29. (b) Autodesk Revit 2027.

Figure 11: User interfaces of the BIM authoring platforms used in BIMArena.

Table 9: Action space used by the computer-use agents in BIMArena.

Category	Action	Description
Mouse Interaction	`mouse_move_to`	Move the cursor to a specified screen coordinate.
	`mouse_click`	Perform a mouse click at the current or specified position.
	`mouse_double_click`	Perform a double-click operation.
	`mouse_press_hold`	Press and hold the mouse button, primarily for flyout interactions.
	`shift_hover`, `commit_select`, `shift_click`	Perform software-specific selection gestures for overlapping or embedded elements.
Keyboard Interaction	`type`	Enter text or numerical values.
	`select_all`	Select all editable content in the active field.
	`press_enter`	Confirm the current input or operation.
	`press_esc`	Cancel or exit the current operation.
	`press_tab`	Navigate between interface elements or candidate selections.
	`delete_selected`	Delete the currently selected element or content.
	`hotkey`	Execute an agent-composed keyboard shortcut or key combination; no predefined shortcut set is imposed.
Navigation	`scroll`	Scroll within the active view or interface region.
External Support	`documentation_retrieval`	Retrieve relevant official software documentation using keyword-based matching.
	`operational_skill_retrieval`	Retrieve capability-aligned operational skills using RAG.

C.2 Software Setup

As summarized in Table 5, we use Graphisoft Archicad 29 and Autodesk Revit 2027 on Windows. To enable programmatic access to the internal BIM model state, we additionally integrate software-specific add-ons for both applications.

Archicad. For Graphisoft Archicad 29, we use the official JSON interface (Graphisoft, 2025b) for programmatic access to project information. To retrieve a more comprehensive representation of the BIM model, we additionally load the open-source Tapir add-on (Version 1.5.3) (ENZYME-APD, 2026), which extends the official interface with additional JSON commands.

Revit. For Autodesk Revit 2027, we implement a lightweight native add-in in C# using the Revit 2027 API (Autodesk, 2026b). The add-in exposes a lightweight local HTTP API for model-state extraction. Its output follows the same representation schema used for Archicad, enabling a unified programmatic evaluator across both applications.

C.3 Action Space and Observation Space

The action space used in BIMArena is summarized in Table 9. In addition to standard GUI interaction primitives for mouse and keyboard, the support setting additionally provides two optional retrieval actions: official software documentation retrieval and operational skill retrieval. Documentation is retrieved through keyword-based matching, whereas operational skills are retrieved using retrieval-augmented generation (RAG) (Lewis et al., 2020).

Visual observations are entirely screenshot-based. At each observation step, the complete computer screen is captured and provided to the agent as visual input. The system automatically retains both the current screenshot and the immediately preceding screenshot, allowing the agent to compare consecutive visual states during execution. All experiments use a fixed resolution of 1920×1080 pixels with a 16:9 aspect ratio to maintain a consistent visual environment across evaluation runs.

C.4 Automated Evaluation Framework

As described in Section 4.4, BIMArena uses a deterministic programmatic verifier rather than human or language-model judges.

Model-State Extraction. After each run, its final model state S_T is extracted through the corresponding software API. The extracted information is converted into a unified schema containing element geometry, types, semantic properties, stories, openings, materials, and composite structures.

Task-Specific Evaluation Constraints. Each task contains a machine-readable set of task-specific evaluation constraints (see Listing 1) defining the required model changes and the elements or properties that must remain unaffected. These constraints are converted into programmatic checkpoints and evaluated against the extracted final model state S_T. Geometric properties are evaluated using task-specific tolerances, while discrete attributes such as element counts and names are checked

exactly. Before inclusion in the benchmark, the expert-constructed target model must satisfy all task-specific evaluation constraints.

Metrics. For each task i, the task-specific evaluation constraints are converted into a set of goal checkpoints $C_i = \{c_{i,1}, \ldots, c_{i,n_i}\}$. Each checkpoint represents one required condition of the target BIM model, such as the existence, geometry, type, or property of an element. Preservation checkpoints P_i are derived from the initial BIM model using the same checkpoint representation and specify the existing elements and properties that must remain unchanged. Given the final model state $S_{T,i}$, let $C_i^* \subseteq C_i$ and $P_i^* \subseteq P_i$ denote the goal and preservation checkpoints satisfied by the final model. For N evaluated tasks, we compute the following metrics.

- **Partial Completion Score (PCS).** PCS measures the fraction of goal checkpoints satisfied for each task:
$$\mathrm{PCS}_i = \frac{|C_i^*|}{|C_i|}.$$
It captures how much of the required target state has been achieved, even when the task is not fully completed. The reported PCS is the average across all tasks:
$$\mathrm{PCS} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{PCS}_i \times 100\%.$$
- **Collateral-Free Rate (CFR).** CFR measures whether all preservation checkpoints remain satisfied:
$$\mathrm{CFR}_i = \mathbf{1}[P_i^* = P_i]\,,$$
where $\mathbf{1}[\cdot]$ equals 1 when the condition holds and 0 otherwise. Thus, a task receives a value of 1 only when all elements and properties that should remain unchanged are preserved. The reported CFR is the average across all tasks:
$$\mathrm{CFR} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{CFR}_i \times 100\%.$$
- **Success Rate (SR).** A task is considered successful only when all goal and preservation checkpoints are satisfied:
$$\mathrm{SR}_i = \mathbf{1}[C_i^* = C_i \wedge P_i^* = P_i]\,.$$
This requires both complete achievement of the target state and preservation of all unaffected model content. The reported SR is the average across all tasks:
$$\mathrm{SR} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{SR}_i \times 100\%.$$

C.5 Validity of the Evaluator

We further validate the evaluator using two reference states: the unchanged initial BIM model and the expert-constructed target BIM model. The analysis is based on the task-specific evaluation constraints and evaluator outputs without re-executing any agent runs.

No-Operation and Expert-Target Baselines. Goal checkpoints C are derived from the task-specific evaluation constraints and are used to compute PCS. Preservation checkpoints P are derived from the initial BIM model and are evaluated through CFR. Here, $|C|$ denotes the number of goal checkpoints, while $|P|$ denotes the number of preservation checkpoints.

As shown in Table 10, an unchanged initial model satisfies none of the goal checkpoints across all three tracks. It therefore obtains 0% PCS and 0% SR, while achieving 100% CFR because no existing elements are modified. In contrast, the expert-constructed target models satisfy all goal and preservation checkpoints and therefore achieve 100% PCS, SR, and CFR across all tracks. These two baselines confirm that the evaluator assigns no task progress to an unchanged model while fully recognizing the intended expert solution.

Table 10: Checkpoint composition and evaluator performance on the unchanged initial model and expert-constructed target model.

			Initial Model			Expert Target Model		
Track	$\|C\|$	$\|P\|$	PCS	SR	CFR	PCS	SR	CFR
Reasoning	1617	1833	0 %	0 %	100 %	100 %	100 %	100 %
Long-Sequence	1617	1833	0 %	0 %	100 %	100 %	100 %	100 %
Atomic	238	652	0 %	0 %	100 %	100 %	100 %	100 %

D Model Access

D.1 Models

We evaluate nine proprietary multimodal models from several major model providers, together with two open-source computer-use agents. The proprietary models are accessed through their corresponding APIs, whereas EvoCUA-32B and OpenCUA-32B are self-hosted. The evaluated models and agents are:

- `gpt-5.6-sol`
- `gpt-5.6-terra`
- `gpt-5.6-luna`
- `claude-opus-5`
- `claude-sonnet-5`
- `gemini-3.7-flash`
- `qwen3.8-max`
- `qwen3.7-plus`
- `muse-spark-1.1`
- `EvoCUA-32B`
- `OpenCUA-32B`

Model Inference Settings. For the proprietary models, we use the provider-supported default sampling configuration. Where configurable, temperature is set to $T = 1.0$, and models with explicit reasoning controls use the high reasoning setting. The maximum response length is 8,192 tokens, and all proprietary models receive the same action declarations through their native function-calling interfaces. EvoCUA-32B and OpenCUA-32B are evaluated using their released computer-use configurations. We retain their original visual-history settings: EvoCUA-32B uses five screenshots (four historical observations plus the current one), while OpenCUA-32B uses three screenshots following its OSWorld configuration. Due to evaluation cost, we perform a single run per model or agent for each task and experimental setting.

Coordinate Frames of Click Outputs. The evaluated models use different coordinate conventions for click targets. GPT-5.6 and Claude return pixel coordinates in the transmitted image frame, whereas Gemini 3.7 Flash, Qwen, and Muse Spark 1.1 use normalized coordinates. EvoCUA-32B and OpenCUA-32B use their native coordinate and image-preprocessing pipelines. We therefore apply model-specific decoding before mapping all predicted clicks back to the native 1920×1080 screen.

Operational Support Compatibility. The retrieval-based operational support is not applied to EvoCUA-32B or OpenCUA-32B. In our main agent interface, support is provided through callable tools, which the agent may invoke alongside the GUI actions. EvoCUA-32B and OpenCUA-32B instead use fixed computer-use action spaces and do not expose a compatible function-calling interface

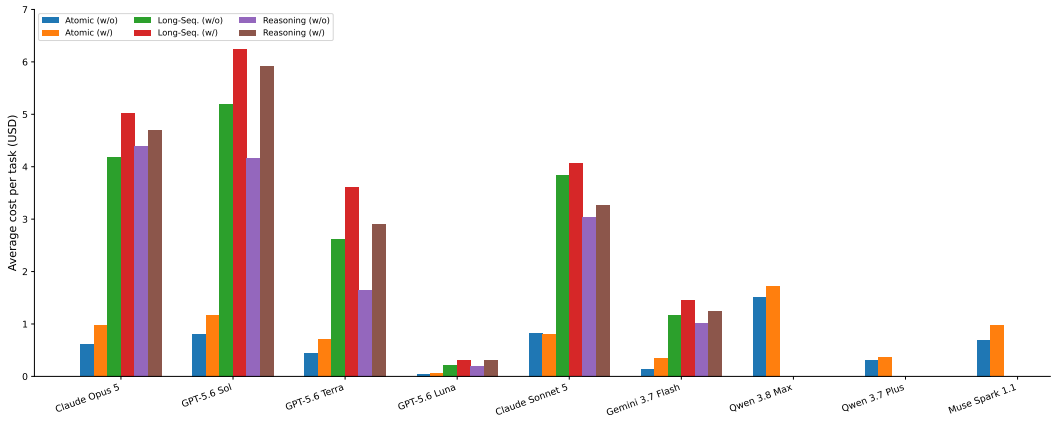


Figure 12: Average API cost per task across BIMArena evaluation tracks and support settings. Models evaluated only on Atomic tasks are shown with Atomic costs only.

for these retrieval tools. Injecting retrieved information externally into their prompts would constitute a different support mechanism and would not be directly comparable. We therefore evaluate both agents only under their native configurations.

D.2 Cost Analysis

Figure 12 reports the average API cost per task across models, task tracks, and support settings. Atomic tasks are consistently the least expensive, while Long-Sequence tasks generally incur the highest cost due to longer interaction trajectories. Reasoning tasks are often slightly cheaper, likely because agents fail to infer the intended objective or terminate earlier on difficult domain-grounded inputs. Costs also vary substantially across models. GPT-5.6 Sol incurs the highest evaluation cost, reaching USD 6.24 per Long-Sequence task and USD 5.92 per Reasoning task with operational support. This is largely attributable to its substantially higher reasoning-token usage, as the model frequently consumes a large fraction of the available reasoning budget during long-horizon execution. In contrast, GPT-5.6 Luna remains below USD 0.32 per task across all tracks. Across all models, task tracks, and support settings, the complete experimental campaign incurred an overall API cost of approximately USD 12K.

D.3 Cross-Software Performance

We further compare performance between Archicad and Revit using the six models evaluated on all three tracks. As shown in Figure 13, Archicad performs better on Atomic tasks (73.9% vs. 66.9% without support; 81.7% vs. 74.0% with support) and Long-Sequence tasks (33.3% vs. 28.5%; 38.3% vs. 33.1%). The gap narrows on Reasoning tasks, where performance is comparable across the two platforms. Operational support improves Atomic and Long-Sequence performance on both platforms, while its effect on Reasoning is smaller and less consistent.

The stronger Archicad performance on Atomic and Long-Sequence tasks may reflect differences in software-specific interaction complexity, including how modeling commands, element settings, and editing operations are exposed through the interface. Such differences may affect the reliability of task execution across both basic operations and longer workflows. In contrast, Reasoning tasks introduce an additional bottleneck in interpreting architectural drawings and domain requirements before execution. At this level, reasoning and target-state inference appear to dominate software-specific interaction differences, which may explain why the performance gap between Archicad and Revit disappears. The similar benefit of operational support across both platforms further suggests that procedural knowledge can reduce part of the interaction burden but does not eliminate the underlying difficulty of deeper reasoning tasks.

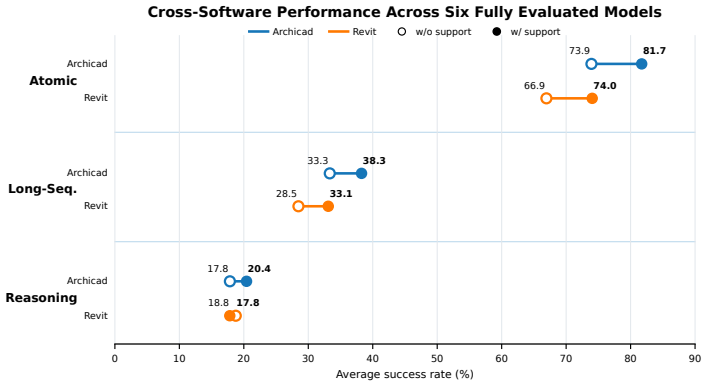


Figure 13: Cross-software performance on Archicad and Revit, averaged over the six models evaluated on all three tracks. Hollow and filled markers denote results without and with operational support, respectively.

E Failure Analysis

As described in Section 5.2, our analysis of failed trajectories reveals four predominant error categories: UI grounding inaccuracies, model geometry understanding, operation planning, and domain knowledge grounding. Here, we present and discuss several representative examples of these failure modes.

UI Grounding Inaccuracy. Figure 14 shows a representative UI grounding failure. Unlike conventional GUI tasks that primarily require locating interface controls, BIM authoring also requires accurately grounding and selecting model elements within the drawing canvas. In this example, the task requires selecting an existing window, replacing it with a triple window type, and changing its width to $2500\,\mathrm{mm}$. However, the agent repeatedly fails to select the target window and attempts several alternative interaction strategies. It eventually moves the window instead of changing its type and subsequently introduces unintended modifications to other elements while attempting to recover. The resulting model therefore differs substantially from the expected result shown in Figure 14.

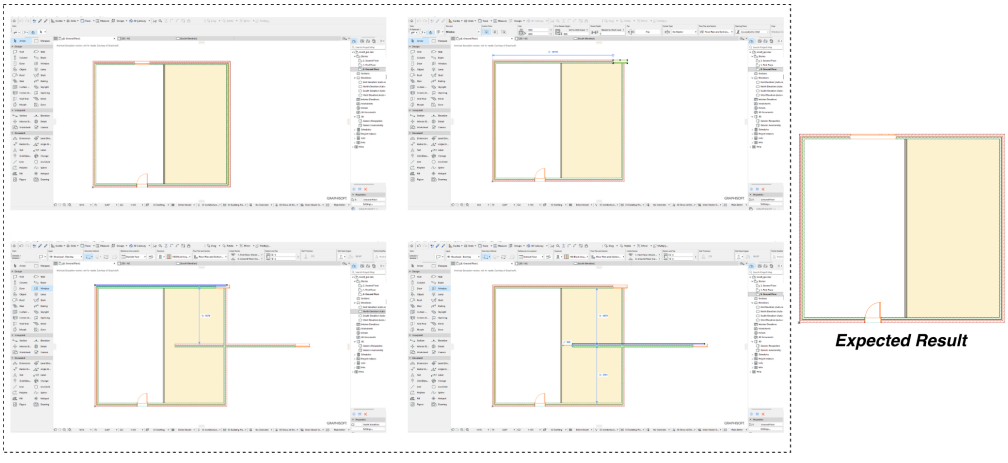


Figure 14: Example of a UI grounding failure. The agent repeatedly fails to select the target window and consequently performs unintended modifications.

Model Geometry Understanding. Figure 15 shows a representative model geometry understanding failure. The task requires placing six $1500 \times 1500\,\mathrm{mm}$ windows on the east exterior wall of

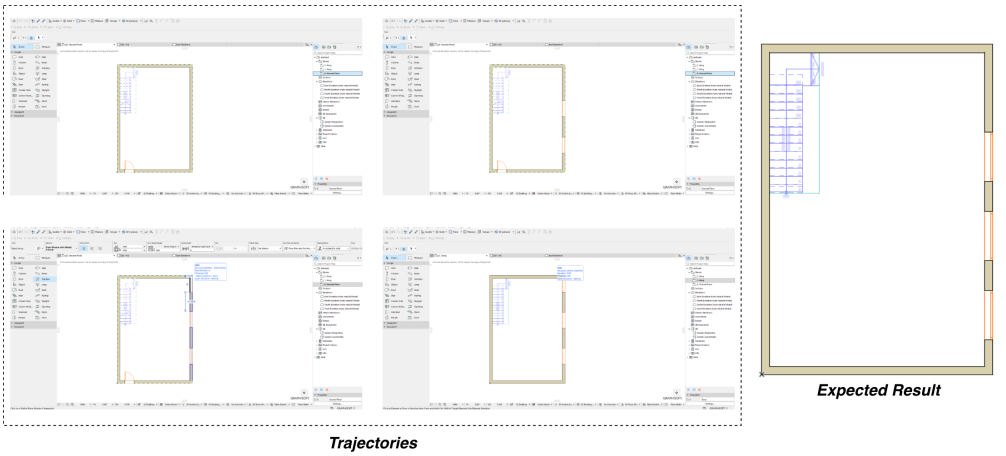


Figure 15: Example of a model geometry understanding failure. The agent misinterprets the visible wall extent and consequently places all windows at incorrect positions.

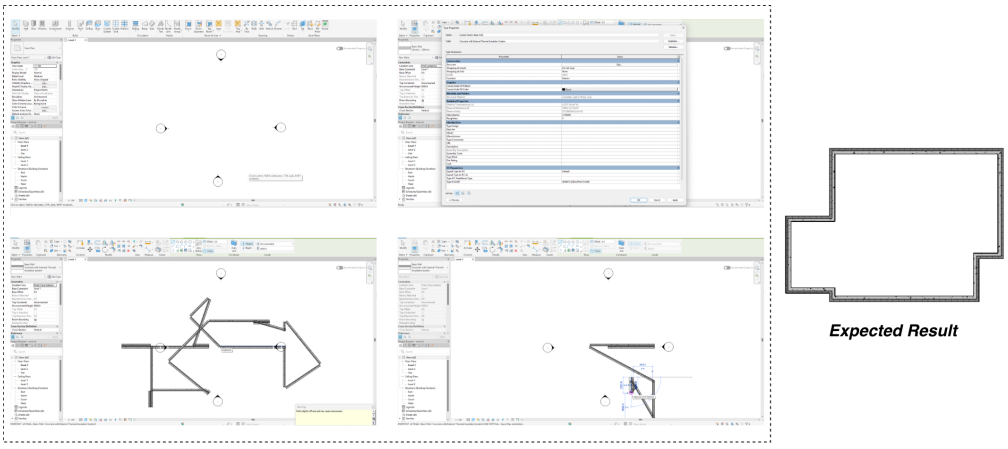


Figure 16: Example of an operation-planning failure. The agent repeatedly retries the same unsuccessful wall-creation strategy instead of diagnosing and adapting its plan.

a two-story building, with three windows on each story at specified vertical positions and story-specific sill heights. The agent correctly identifies the required window size, sill heights, and number of windows, but incorrectly interprets the visible dimension chain as the full wall extent of $7200\,\mathrm{mm}$, whereas the actual wall length is $10000\,\mathrm{mm}$. It therefore estimates the window positions from screen-space proportions instead of using the specified distances. As a result, all windows are systematically misplaced, and none of the six required placement units is satisfied despite otherwise correct window properties, as can be seen by comparison with the expected result shown in Figure 15.

Operation Planning. The task requires first creating a four-layer exterior wall composite and then constructing a closed exterior loop consisting of ten wall segments using the specified coordinates. As shown in Figure 16, the agent successfully creates the required wall type within the first 110 interaction rounds but repeatedly fails during wall placement. After each attempt produces incorrectly extended wall segments, the agent does not diagnose the underlying cause and instead undoes the entire wall chain and repeats essentially the same procedure. Consequently, none of the required wall-placement units is completed despite the correct wall composite being created. This illustrates a failure to adapt the execution strategy after repeated unsuccessful outcomes, as reflected by the substantial discrepancy between the final model and the expected result shown in Figure 16.



Figure 17: Example of a domain knowledge grounding failure. The agent identifies that the bedroom layout requires adjustment but fails to infer and realize the correct domain-specialized target.

Domain Knowledge Grounding. The task requires assessing whether two bedrooms in a three-person dwelling have reasonable dimensions and resizing undersized rooms by moving interior walls. According to the referenced design requirements (Neufert & Neufert, 2012), the child's bedroom should provide at least $2600 \times 3100\,\mathrm{mm}$ clear space, while the parents' bedroom should provide at least $3600 \times 3500\,\mathrm{mm}$. In the paired Long-Sequence task, these requirements are stated explicitly and the agent successfully completes the corresponding wall modifications, indicating that the required software operations can be executed when the target is directly specified. In the Reasoning task, however, the agent must infer the appropriate room dimensions from the domain context itself. Without explicit dimensional requirements, the agent must independently determine whether the existing room dimensions are acceptable and identify any necessary modifications. As shown in Figure 17, the agent recognizes that the existing room sizes are inadequate but derives an incorrect target and introduces unnecessary modifications and extra elements. The substantial discrepancy from the expected result therefore reflects a failure in domain knowledge grounding rather than the underlying wall-editing operation.

F AGENT PROMPT

The complete system prompt used for all BIMArena agent runs is provided below. The same prompt is used across all evaluated models and task tracks, while task-specific instructions and additional context are provided separately for each case.

Listing 4: System prompt used for BIMArena agent evaluation.

```
You are a computer-use agent operating a BIM authoring application (Autodesk Revit or
    Graphisoft Archicad) through its graphical user interface. Your job is to bring the BIM
    model to the state required by the task - given as a written specification and/or
    attached architectural drawings - using only mouse and keyboard actions on the screen.
    Nothing else observes the model or checks your work.

=== HOW THE LOOP WORKS ===

- Each turn you may issue one or more actions from the action space; they execute in order.
    The run is limited by TURNS, not by actions. Batch every action you can already commit to
     into one turn; spend a turn on a single action only when the next step depends on
    something you have not yet seen. One action per turn exhausts the budget half-done.
- The run ends when you reply without actions. Do not close the project or the application.

=== WRITE YOUR REASONING ===

Every turn that issues actions starts with a short written rationale: what the last screenshot
     showed, what you will do next and WHY (name the alternative you rejected when there was
    a real choice), and any value you JUDGED rather than read from the requirement. This
    record is the only reasoning that survives into later turns. A decision you do not write
```

```
    down cannot be checked later; an error whose cause you do not write down will be repeated
    .

=== ORIENT FIRST ===

Identify which application is in front of you from the screenshot, the active view and story,
    the display units, and what already exists in the model. Drive only that application for
    the whole run; its interface, terminology and workflows differ from the other one.

=== WORKING IN THE APPLICATION ===

- CHECK WHAT ALREADY EXISTS before creating anything. Never create a duplicate of an element
    that is already there; if the task asks to change something, modify that element.
- Use only types, composites, materials and library elements that actually exist in the
    project or its library: open the type selector, library or attribute dialog and read the
    exact name. Never type a name you have not seen on screen. If the requirement names
    something unavailable, choose the closest existing option BY FUNCTION and record the
    substitution in your rationale.
- Enter geometry numerically: type coordinates, lengths and offsets into the application's
    input fields (info box, temporary dimensions, properties panel) and confirm with Enter.
    Snap to existing geometry for connections; zoom in when a snap target is ambiguous.
- Order dependencies: stories before elements placed on them, walls before the doors and
    windows they host, slabs before what stands on them.
- Leave unrelated parts of the model untouched. Do not move, delete or retype elements the
    task does not mention.
- When something goes wrong (an error dialog, an element in the wrong place, a tool that does
    not respond): observe, read the dialog or the resulting state, identify the cause, and
    fix THAT cause. Prefer Undo (Ctrl+Z) to revert a wrong step over deleting and rebuilding.
     Do NOT repeat the identical action sequence and do not switch to guessing.
- Work in MILLIMETRES throughout. A drawing dimension is used as-is; never convert to metres.
    If the project's display units differ, account for it consistently and note it.


=== OPERATIONAL SUPPORT (only when retrieval actions are available) ===

- `operational_skill_retrieval` returns a step-by-step procedure for a specific capability in
    the current application. Retrieve it when needed; it also specifies the required
    reference line, location line, and draw mode.
- `documentation_retrieval` returns official documentation pages. Use it only for specific
    questions, such as an unfamiliar dialog or a setting you cannot locate. Read the
    retrieved text critically, as it may describe functions, options, or workflows that are
    irrelevant to the current task. Do not perform operations that the task does not require
    simply because they are mentioned in the documentation.


=== FINISHING ===

Before you stop, verify your own work: observe the model you actually built (plan view, and
    the 3D view where it helps) and compare it against the requirement - element counts,
    positions, types, stories, openings. Fix what is clearly wrong: missing elements, wrong
    placement, wrong type, unintended changes.

Then reply with a plain summary: what you created or changed, any value you judged rather than
     read, any substitution you made, and anything you could not complete and why. That reply
     ends the run.

That final reply always BEGINS with one line, then the summary:
 RESULT: completed
 RESULT: abandoned - <one sentence: what you could not do and why>
```