# ENGIWORLD: WHAT CAN FRONTIER AGENTS DELIVER IN PROFESSIONAL ENGINEERING ENVIRONMENTS?

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Autonomous agents have made rapid progress in general-purpose computer use, but reliable automation of professional industrial engineering remains out of reach, as engineering workflows demand reasoning over geometric and physical constraints and dependencies preserved across software and design stages. We present **ENGIWORLD**, the first benchmark structured around the complete design loop: 1,301 expert-curated tasks spanning 6 engineering domains (CAD, CAE, CAM, BIM, EDA, and 3D visualization) and 26 professional software platforms, with both GUI and CLI interfaces and 6 task types ranging from software-selection to open-ended tasks. We further introduce an artifact-centric evaluation methodology built on a unified domain-verifier suite, which programmatically checks the geometric validity, physical feasibility, and rule compliance of final and intermediate artifacts, and scores quantitative design tasks continuously by specification attainment rather than binary success. Evaluation of seven frontier models reveals a substantial capability gap: the strongest model achieves an EngiScore of only 44.3, and just 3.6% of multi-software attempts succeed. ENGIWORLD provides the first rigorous foundation for measuring progress toward agents that operate professional engineering software end to end.

## 1 INTRODUCTION

Large language models (LLMs), multimodal large language models (MLLMs), and agents built on them have achieved transformative progress on general computer-use tasks, and can now plan, perceive, and act across digital environments spanning web navigation, office productivity, and software engineering Deng et al. (2023); Zhou et al. (2023); Xie et al. (2024); Rawles et al. (2024); Kapoor et al. (2024); Jimenez et al. (2023); Yang et al. (2023); Huang et al. (2024); Lai et al. (2023). However, their potential in professional industrial design software, a domain of enormous economic and societal value, remains largely unexplored Ren et al. (2025); Gao et al. (2025); Liu et al. (2026).

Although existing benchmarks have succeeded in evaluating general digital workflows, they expose significant limitations when applied to industrial domains. First, benchmarks such as OSWorld (Xie et al., 2024), Spider2-V (Cao et al., 2024), and ScienceBoard (Sun et al., 2026a) determine success by comparing final states or reference solutions, which cannot verify the functional integrity of engineering artifacts: a visually "correct" model may be unusable due to non-manifold geometry, improper topological constraints, or violations of physical laws Wu et al. (2021); Jayaraman et al. (2021); Li et al. (2022); although FEABench (Mudur et al., 2025), BIM-Edit (Nithyanantham et al., 2026), and CADTestBench (Mallis et al., 2026) have begun to validate artifact quality, each is confined to a single software platform and a single task format. Second, existing work lacks sufficient interaction depth: Text2CAD (Khan et al., 2024) is dominated by single-turn generation, and although CADWorld (Dong et al., 2026) introduces GUI interaction, its step budget is capped at only one hundred steps, whereas industrial tasks require maintaining parametric logic over long operation sequences, where an early error can invalidate all subsequent operations Man et al. (2025); Gong et al. (2026). Finally, current frameworks suffer from a significant *feedback gap*: agents receive only textual signals (e.g., compilation errors) (Wu et al., 2024; Yue et al., 2025), and GUI-EDA (Li et al., 2025) even resorts to offline single-step prediction without executing any action, whereas engineers rely on native feedback from domain tools, such as rendered outputs, simulation convergence curves, and design-rule violation reports, to iteratively refine their designs. However, real engineering work is a complete **design loop**, yet each of these benchmarks covers only one link of it.

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.

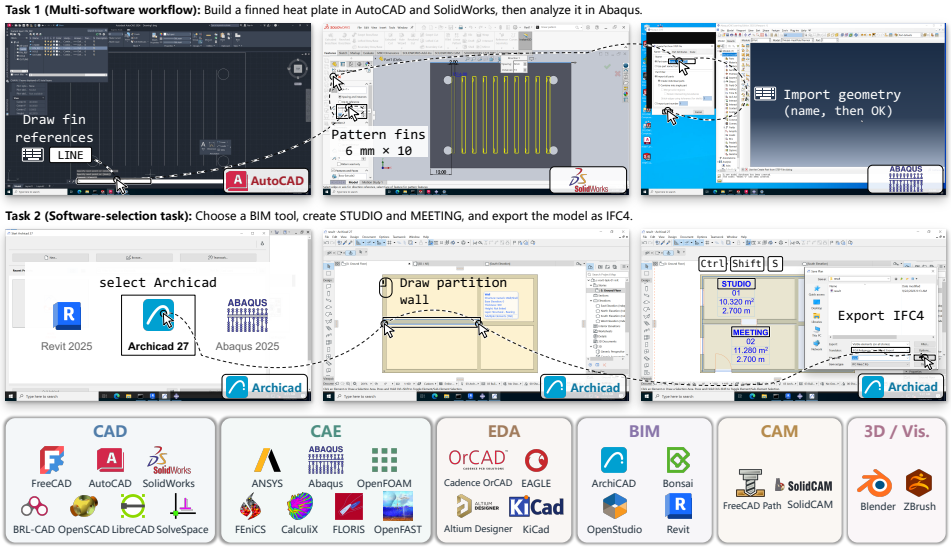



Figure 1: **Overview of ENGIWORLD.** The benchmark covers 26 software platforms and workbenches across six engineering domains, and includes six task types that together constitute an evaluation of the complete design loop.

To bridge this gap, we introduce **ENGIWORLD**, the first benchmark structured around the complete design loop for evaluating agents in professional industrial design environments. As shown in Figure 1, ENGIWORLD spans six engineering domains (CAD, CAE, CAM, BIM, EDA, and 3D visualization), covering 26 software platforms and workbenches with 1,301 expert-curated tasks; the tasks are grounded in real engineering processes (Gao et al., 2025) and require agents to interact with real professional software over multiple turns through GUI or CLI interfaces, maintaining parametric logic across long sequences of operations and applications (Man et al., 2025; Gong et al., 2026). Six task types each evaluate one dimension of the engineering capability stack: *image-based modeling*, *software selection*, *single-software and long-horizon execution*, *quantitative design*, *multi-software coordination*, and *open-ended tasks* that provide only the requirements and a blank computer, together covering all stages of the design process (Figure 3).

Furthermore, we propose an **artifact-centric evaluation methodology**. To address the open and non-unique nature of industrial artifacts, we extend artifact-level programmatic verification from single-domain precedents (Mudur et al., 2025; Nithyanantham et al., 2026; Mallis et al., 2026; Singh et al., 2026) into a unified domain-verifier suite spanning all six domains, which checks the geometric accuracy, physical feasibility, and rule compliance of both final and intermediate artifacts (e.g., STEP files, netlists, and G-code) (Wu et al., 2024; Yue et al., 2025); unlike the binary judgments of existing benchmarks, quantitative design tasks are gated by feasibility checks and report continuous quality scores according to the degree of specification attainment, evaluating *how well* a task is completed rather than merely *whether* it is completed.

We extensively evaluate seven state-of-the-art agents, covering three proprietary models and four open-source models. The results show that even frontier models fall significantly short of industrial-grade engineering standards. Diagnostic analysis reveals failure modes across every stage of the design loop: agents struggle to coordinate cross-application dependencies, satisfy precise geometric and physical constraints, and verify deliverables before completion, and they frequently overlook critical feedback such as solver residuals. These findings highlight the profound gap between general intelligence and the demands of professional engineering, positioning ENGIWORLD as a rigorous foundation for future research on industrial agents.

## 2 RELATED WORK

**Computer-Use Benchmarks.** Benchmarks for autonomous agents cover software engineering (Jimenez et al., 2023; Yang et al., 2023; Huang et al., 2024), web interaction (Deng et al., 2023; Zhou et al., 2023; Koh et al., 2024), and desktop operation (Xie et al., 2024). OSWorld evaluates task completion in real desktop environments, while Spider2-V (Cao et al., 2024) and

ScienceBoard (Sun et al., 2026a) extend evaluation to professional data-engineering and scientific workflows. These settings combine interaction through graphical or command-line interfaces with execution-based outcome checks. More recent benchmarks examine long-horizon execution and workflow dependencies: OSWorld2 (Yuan et al., 2026) studies extended computer-use tasks, and Agents' Last Exam (ALE) (Sun et al., 2026b) evaluates professional workflows with verifiable deliverables across industries, including engineering and 3D applications.

Table 1: **Comparison with existing benchmarks.** *Cross*: cross-software workflows; *Loop*: execution feedback; *Choice*: software selection; *Objective*: engineering performance targets. ALE* is a subset of ALE comprising 25 engineering and 3D tasks.

| Benchmark | Size | Platforms | Domain | Interface | Cross | Loop | Choice | Objective |
|---|---|---|---|---|---|---|---|---|
| Text2CAD [17] | 170k models | – | CAD | Text2Artifact | ✗ | ✗ | ✗ | ✗ |
| VerilogEval [22] | 156 tasks | – | EDA | Code Generation | ✗ | ✗ | ✗ | ✗ |
| RTLLM [25] | 30 designs | – | EDA | Code Generation | ✗ | ✗ | ✗ | ✓ |
| OSWorld [39] | 369 tasks | 9 | General Desktop | GUI+CLI | ✓ | ✓ | ✗ | ✗ |
| Spider2-V [3] | 494 tasks | 20 | Data Engineering | GUI+CLI | ✓ | ✓ | ✗ | ✗ |
| ScienceBoard [35] | 169 tasks | 6 | Scientific | GUI+CLI | ✓ | ✓ | ✗ | ✗ |
| VideoCAD [27] | 41k videos | 1 | CAD | GUI | ✗ | ✗ | ✗ | ✗ |
| CADWorld [7] | 200 tasks | 1 | CAD/CAE/CAM | GUI | ✗ | ✓ | ✗ | ✗ |
| CADTestBench [26] | 200 designs | 1 | CAD | CLI | ✗ | ✓ | ✗ | ✗ |
| FEABench [29] | 215 tasks | 1 | CAE | CLI | ✗ | ✓ | ✗ | ✗ |
| GUI-EDA [21] | 2,082 steps | 5 | EDA/CAE | GUI | ✗ | ✗ | ✗ | ✗ |
| BIM-Edit [30] | 324 tasks | 1 | BIM | CLI | ✗ | ✓ | ✗ | ✗ |
| BlenderGym [11] | 245 pairs | 1 | 3D Graphics | CLI | ✗ | ✓ | ✗ | ✗ |
| ALE* [36] | 25 tasks | 13 | Engineering/3D | GUI+CLI | ✓ | ✓ | ✗ | ✗ |
| **EngiWorld** | **1,301 tasks** | **26** | **CAD/CAE/CAM/ EDA/BIM/3D** | **GUI+CLI** | ✓ | ✓ | ✓ | ✓ |

**Engineering Agents and Benchmarks.** Engineering automation encompasses both structured artifact generation and interactive software use. DeepCAD (Wu et al., 2021) and Text2CAD (Khan et al., 2024) generate parametric CAD construction sequences, while RTLCoder (Liu et al., 2024) generates hardware descriptions. VerilogEval (Liu et al., 2023) assesses functional correctness through simulation, and RTLLM (Lu et al., 2024) additionally evaluates power, performance, and area objectives. Tool-using approaches incorporate planning and execution feedback into CAD modeling, electronic design, and numerical simulation (Gong et al., 2026; Wu et al., 2024; Chen et al., 2024; Yue et al., 2025). VideoCAD (Man et al., 2025) and GUI-EDA (Li et al., 2025) provide datasets for learning and evaluating professional GUI interactions and action prediction.

Engineering benchmarks also assess whether generated artifacts satisfy domain-specific requirements. CADWorld (Dong et al., 2026) evaluates GUI-based FreeCAD workflows spanning modeling, simulation, and manufacturing. CADTestBench (Mallis et al., 2026) uses executable tests to check geometric and topological requirements, while CADEngBench (Singh et al., 2026) examines engineering-oriented CAD capabilities. Beyond CAD, FEABench (Mudur et al., 2025) evaluates multiphysics problem solving in COMSOL, BIM-Edit (Nithyanantham et al., 2026) assesses IFC model edits through geometric, semantic, and topological criteria, and BlenderGym (Gu et al., 2025) evaluates graphics editing through start–goal scene pairs. Across these benchmarks, evaluation ranges from functional simulation and executable constraint checks to geometric and visual comparisons, reflecting the different requirements of engineering artifacts.

## 3 The EngiWorld Benchmark

In this section, we describe the task formulation and interaction model (Section 3.1), the software environments and interaction interfaces (Section 3.2), the task construction pipeline and benchmark coverage (Section 3.3), and the artifact-centric evaluation protocol (Section 3.4).

### 3.1 Benchmark Overview and Task Formulation

An EngiWorld task places an agent in a native engineering software environment and asks it to produce or modify engineering artifacts under a task specification. A specification may include a natural-language objective, reference media, initial project files, and required intermediate or final

deliverables. We represent each task as

$$\tau_i = (g_i, x_i, e_i, \mathcal{A}_i, Y_i, C_i),$$

where $g_i$ is the task objective, $x_i$ contains task inputs such as reference media and initial files, $e_i$ specifies the software environment and workspace state, $\mathcal{A}_i$ is the set of available interface actions, $Y_i$ defines the required deliverables, and $C_i$ contains the acceptance criteria. The formulation mirrors the stages of the design loop introduced in Section 1: $x_i$ and $e_i$ specify how intent and environment are presented to the agent, $\mathcal{A}_i$ governs construction, and $Y_i$ together with $C_i$ encodes specification and delivery. Figure 2 summarizes the interaction and artifact-centric evaluation process.

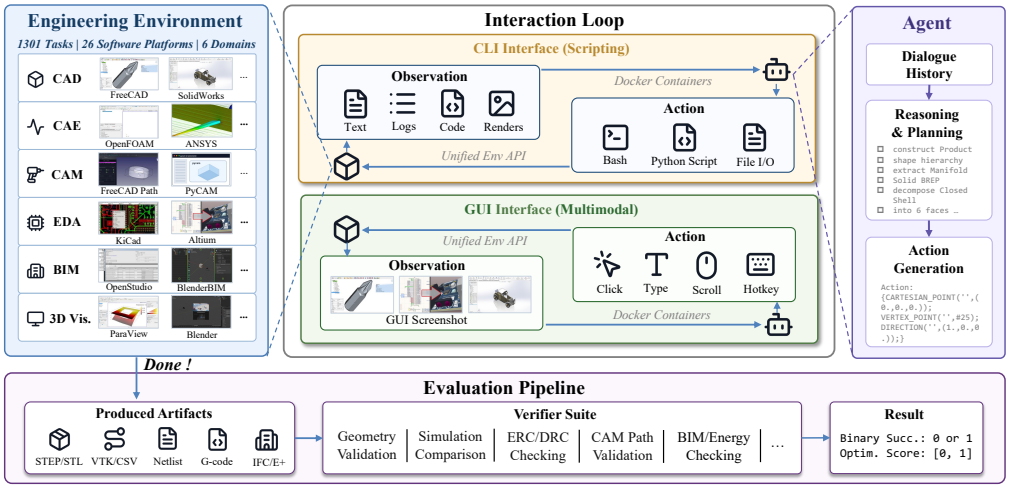



Figure 2: **Overview of the ENGIWORLD framework.** Agents interact with engineering applications through CLI or GUI interfaces and are evaluated from the artifacts produced during execution. Domain-specific verifiers check whether these artifacts satisfy the required engineering constraints.

The interaction is modeled as a partially observable sequential decision process (Kaelbling et al., 1998). Let $s_t \in \mathcal{S}_i$ denote the complete state of the software environment at decision turn $t$, including application state, workspace files, and artifacts generated during execution. The agent receives an observation $o_t \in \mathcal{O}_i$ through the task-specific interface and maintains an interaction history $h_t$. GUI observations consist primarily of rendered application states and may include accessibility information, whereas CLI observations consist of terminal outputs, file contents, and execution logs. Given the task objective and current history, the agent selects

$$a_t \sim \pi_\theta(\cdot \mid g_i, o_t, h_t), \qquad a_t \in \mathcal{A}_i \cup \{\texttt{DONE}, \texttt{FAIL}\}.$$

For interface actions $a_t \in \mathcal{A}_i$, the environment transitions according to

$$s_{t+1} \sim P_i(\cdot \mid s_t, a_t), \qquad o_{t+1} \sim \Omega_i(\cdot \mid s_{t+1}),$$

where $P_i$ and $\Omega_i$ denote the task-specific transition and observation models. The episode terminates when the agent declares `DONE` or `FAIL`, or reaches the task-specific decision limit. This formulation captures the interactive, closed-loop nature of engineering workflows, in which each action changes the software state and determines the feedback available for subsequent decisions.

At termination, evaluation is performed on the submitted artifacts $\mathcal{Y}_T$ rather than on the action trajectory or the final interface state. For binary tasks, each acceptance criterion is represented by an indicator $\phi_{ik}(\mathcal{Y}_T) \in \{0, 1\}$, and a task succeeds only when all required criteria are satisfied:

$$S_i(\mathcal{Y}_T) = \prod_{k=1}^{K_i} \phi_{ik}(\mathcal{Y}_T). \tag{1}$$

Missing, invalid, or unreadable deliverables fail the corresponding criteria. Quantitative design tasks are evaluated in two stages:

$$F_i(\mathcal{Y}_T) = \prod_{k=1}^{K_i} \psi_{ik}(\mathcal{Y}_T), \qquad Q_i(\mathcal{Y}_T) = \begin{cases} q_i(\mathcal{Y}_T), & F_i(\mathcal{Y}_T) = 1, \\ 0, & F_i(\mathcal{Y}_T) = 0, \end{cases} \tag{2}$$

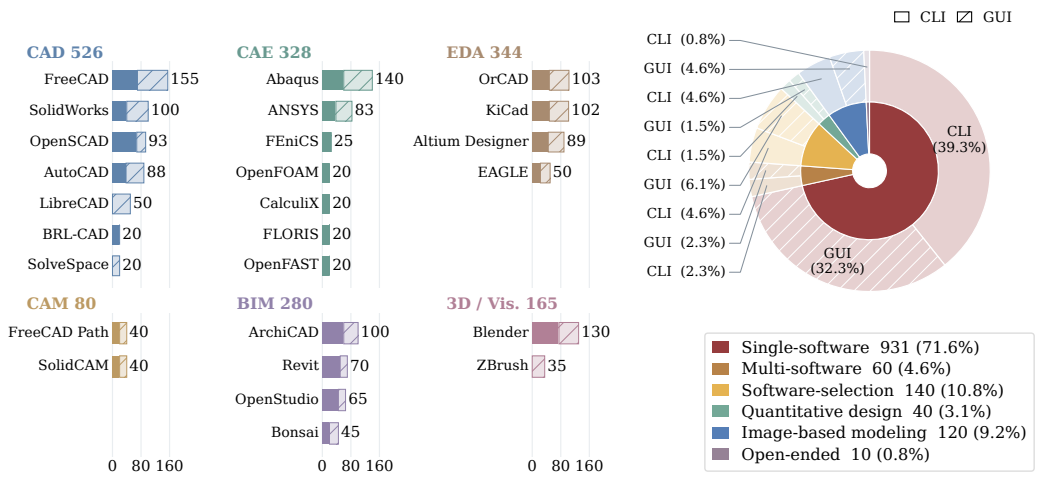
where $\psi_{ik}$ verifies an engineering constraint and $q_i$ measures task-specific design quality. Binary tasks therefore require complete satisfaction of their acceptance criteria, while quantitative tasks receive a continuous quality score only when the produced design is feasible.

### 3.2 Environments and Interaction

EngiWorld tasks run in isolated Windows or Ubuntu environments containing the required native engineering applications. Each episode starts from a task-specific workspace state with the necessary input files and application documents prepared. Single-software tasks use a designated application, whereas Multi-software tasks provide a shared workspace in which intermediate artifacts are transferred across applications and checked at subsequent stages. Software-selection tasks allow the agent to choose from a permitted set of applications under task-defined interface constraints.

The benchmark supports two interaction interfaces. With the GUI interface, agents operate applications through visual observations and mouse or keyboard actions. With the CLI interface, agents use terminal commands and scripts and receive textual feedback from the software and operating system. The two interfaces expose different observation and action spaces but share the same artifact-based evaluation protocol, so performance is determined by the validity of the resulting engineering deliverables rather than by the interaction modality.

### 3.3 Task Construction and Coverage




**(a) Software coverage.** **(b) Task types and interfaces.**

Figure 3: **Dataset composition.** EngiWorld contains 1,301 tasks across six engineering domains and 26 software platforms and workbenches. Software coverage counts task–software associations, including all applications in a workflow and all permitted tools in software-selection tasks.

We construct EngiWorld through an expert-calibrated, LLM-assisted pipeline. Domain experts identify representative workflows from software documentation, tutorials, and other engineering resources, and convert them into executable task specifications with task instructions, initial files, reference artifacts, required deliverables, and task-specific verifiers. Experts review sampled specifications and executions for correctness, completeness, ambiguity, and evaluability. Every final task is then executed and reviewed in its designated software environment. Detailed construction and validation procedures are provided in App. A.

EngiWorld contains 1,301 task instances across six engineering domains and 26 software platforms and workbenches. The benchmark includes 691 tasks using the CLI interface and 610 tasks using the GUI interface (Figure 3b). Because a workflow may involve multiple applications, software coverage is reported using task–software associations in addition to task instances. The benchmark contains 1,723 associations, where each association represents one software involved in a task. Multi-software tasks therefore contribute one association for each participating application, while Software-selection tasks include all permitted candidate applications. This convention captures the

breadth of the engineering ecosystem covered by ENGIWORLD, spanning geometric models, simulation fields, circuit schematics, building models, and 3D scenes (Figure 3a).

The six task types form a disjoint partition of the benchmark. **Single-software tasks** (931) require agents to complete an objective within a designated application. **Multi-software tasks** (60) require intermediate artifacts to be transferred and preserved across multiple applications. **Software-selection tasks** (140) allow agents to choose from a permitted set of tools. **Quantitative design tasks** (40) evaluate both engineering feasibility and a task-specific design objective. **Image-based modeling tasks** (120) require agents to construct artifacts from visual references. **Open-ended tasks** (10) remove predefined software and workflow constraints, allowing agents to select the tools and execution strategies. Together, the six types instantiate successive stages of the design loop: image-based modeling probes intent grounding, software-selection probes tool selection, single-software tasks probe precise construction, quantitative design probes specification attainment, multi-software tasks probe cross-toolchain delivery, and open-ended tasks require agents to bootstrap the environment itself.

### 3.4 ARTIFACT-CENTRIC EVALUATION

Operationalizing the validation stage of the design loop, ENGIWORLD evaluates the engineering artifacts submitted at termination using programmatic verifiers. Each verifier reopens the submitted files, extracts the properties required by the task, and checks them against structural requirements or tolerance-bounded numerical criteria. The checks cover geometric dimensions and topology, simulation quantities, schematic connectivity, building-model entities and relationships, and 3D scene structure. For Multi-software tasks, verifiers additionally inspect intermediate artifacts and consistency across workflow stages. For binary tasks, all required acceptance criteria must pass to obtain $S_i = 1$. For quantitative design tasks, the verifier first checks engineering feasibility through $F_i$ and then computes the quality score $Q_i$ only for feasible outputs; infeasible submissions receive $Q_i = 0$. Missing, invalid, or unreadable deliverables fail the corresponding criteria. Verifiers are tested with reference artifacts, alternative valid solutions, and controlled defective submissions to ensure that they accept valid realizations and detect task-relevant violations. Detailed checker implementations, numerical tolerances, and validation results are provided in App. B.2 and App. B.3.

## 4 EXPERIMENTS

In this section, we present the experimental setup and the main results of seven frontier models on ENGIWORLD (Sections 4.1 and 4.2), followed by ablation studies on observation design and interaction history (Section 4.3) and a failure analysis (Section 4.4).

### 4.1 EXPERIMENTAL SETUP

**Models.** We evaluate seven frontier models: GPT-5.6-Sol (XHigh) (OpenAI, 2026), Claude Opus 5 (Max) (Anthropic, 2026), Kimi K3 (Max) (Moonshot AI, 2026), Gemini 3.7 Flash (High) (Google DeepMind, 2026), Qwen 3.8 Max and Qwen 3.8 Flash (XHigh) (Alibaba Cloud, 2026), and DeepSeek V4.1 Flash (Max) (DeepSeek, 2026). Each model receives the same task specification and tool instructions in a zero-shot setting.

**Task selection.** To control evaluation cost, we evaluate all seven models on the same stratified subset of 300 tasks, comprising 152 CLI-interface tasks and 148 GUI-interface tasks. The subset allocates four or five tasks to each of 73 strata defined by software and task category, while covering all six engineering domains.

**Execution.** Each episode starts from a clean Windows or Ubuntu virtual-machine snapshot with task-specific inputs and application states (App. B.1). Agents using the GUI interface operate through PyAutoGUI with $1920 \times 1080$ screenshots, whereas agents using the CLI interface issue terminal commands and scripts; images can be inspected through `readimg`. For both interfaces, we retain the most recent 15 interaction rounds. The default decision limits are 200 rounds for GUI-interface tasks and 100 for CLI-interface tasks, increased to 300 and 150, respectively, for Multi-software and Open-ended tasks.

**Metrics.** Task-specific verifiers recompute engineering properties from the submitted artifacts. We report **EngiScore**, which aggregates binary task success and quantitative design quality. For an evaluated task set $\mathcal{D}$,

$$\text{EngiScore}(\mathcal{D}) = \frac{100}{|\mathcal{D}|}\left(\sum_{i\in\mathcal{D}\cap\mathcal{T}_{\text{bin}}} S_i + \sum_{i\in\mathcal{D}\cap\mathcal{T}_{\text{quant}}} Q_i\right), \tag{3}$$

where $S_i \in \{0,1\}$ indicates whether all acceptance criteria pass, and $Q_i \in [0,1]$ is the task-specific quality score, with infeasible submissions receiving zero. EngiScore ranges from 0 to 100 and weights every task equally. On binary-only task sets, it is equivalent to the success rate in percentage points. We also report the mean number of interaction steps and mean API cost per task, averaged over all 300 tasks, including unsuccessful runs. Each step corresponds to one decision round, and step counts are capped at the prescribed task-specific limits.

### 4.2 Main Results

Table 2: **Main results on EngiWorld.** EngiScore divided by task category and interface. Steps and Cost are per-task means over all 300 tasks. Higher EngiScore and lower Steps and Cost are preferred; best and second-best values are **bold** and underlined.

| Model | By Task category | | | | | | By Interface | | Overall | Execution | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Single (175) | Multi (24) | Select. (28) | Quant. (33) | Image (36) | Open (4) | CLI (152) | GUI (148) | EngiScore (300) | Steps | Cost ($) |
| Claude Opus 5 | **50.3** | **8.3** | **25.0** | 32.8 | **66.7** | 25.0 | 47.9 | **40.5** | **44.3** | 70.2 | 19.02 |
| GPT-5.6 Sol | 44.6 | 4.2 | 17.9 | **36.4** | 50.0 | 0.0 | 47.1 | 28.6 | 38.0 | **54.9** | 9.73 |
| Qwen3.8 Max | 26.3 | 0.0 | 7.1 | 22.3 | 52.8 | **75.0** | 34.7 | 16.6 | 25.8 | 108.1 | 6.52 |
| DeepSeek V4.1 Flash | 30.3 | **8.3** | 7.1 | 16.2 | 30.6 | **75.0** | **48.3** | 2.0 | 25.5 | 116.2 | 0.81 |
| Gemini 3.7 Flash | 30.9 | 0.0 | 3.6 | 20.6 | 30.6 | **75.0** | 36.1 | 14.2 | 25.3 | 110.8 | 2.11 |
| Kimi K3 | 30.3 | 0.0 | 3.6 | 21.9 | 27.8 | 25.0 | 31.3 | 16.7 | 24.1 | 92.5 | 5.90 |
| Qwen3.8 Flash | 20.0 | 4.2 | 7.1 | 9.9 | 19.4 | 0.0 | 30.8 | 1.0 | 16.1 | 120.5 | **0.45** |

**Overall performance.** Claude Opus 5 achieves the highest overall EngiScore at 44.3, followed by GPT-5.6 Sol at 38.0 (Table 2). The absolute scores remain low: all seven models receive zero scores on 128 of the 300 tasks (Figure C.1). Performance depends strongly on the interface. Claude and GPT perform similarly on the CLI interface, while Claude has an advantage on GUI tasks. DeepSeek obtains the highest CLI score among the remaining models, but succeeds on only three GUI tasks, indicating that strong command-line performance does not translate directly to visual interaction.

**Task types.** Performance drops sharply when a task requires coordination across applications. Claude completes roughly half of the Single-software tasks and two thirds of the Image-based modeling tasks, but only 6 of 168 Multi-software attempts succeed across all models. Software-selection tasks are also difficult, with Claude solving only one quarter of them. These results point to a central limitation of agents: operating an individual application is easier than selecting tools, transferring intermediate artifacts, and preserving dependencies across a multi-stage workflow. In terms of the design loop, current agents can execute individual stages but break down at the transitions between them. Results for Quantitative design and Open-ended tasks are reported in App. C.

**Engineering domains.** The domain results reveal substantial specialization across models (Figure 4). Claude leads in CAD, CAE, EDA, and BIM and ties GPT in CAM, whereas Qwen3.8 Max performs best in 3D visualization. CAM is the most difficult domain for every model. Domain rankings differ from task-type rankings: Qwen leads in 3D visualization despite its lower overall score, while Claude's advantage on Image-based modeling does not extend uniformly across domains.

**Execution cost.** Models average 54.9–120.5 interaction steps per task. GPT uses the fewest steps (54.9), followed by Claude (70.2). DeepSeek and Gemini achieve similar scores of approximately 25, but DeepSeek costs less than half as much. Claude reaches the highest score at nearly twice GPT's API cost.

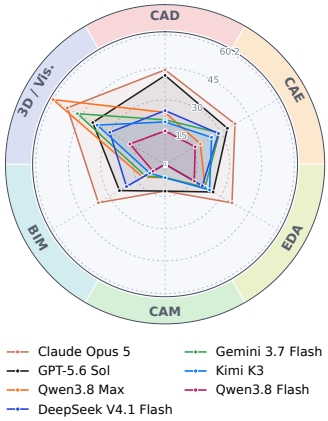



Figure 4: **Performance by domain.** Tasks involving multiple domains contribute to each relevant domain.

| Model | *n* | Failure reason (%) F1 | F2 | F3 | F4 | F5 |
|---|---|---|---|---|---|---|
| Claude Opus 5 | 165 | 73.9 | 19.4 | 1.2 | 5.5 | 0.0 |
| GPT-5.6 Sol | 184 | 87.0 | 13.0 | 0.0 | 0.0 | 0.0 |
| DeepSeek V4.1 Flash | 218 | 24.3 | 62.4 | 0.5 | 12.8 | 0.0 |
| Qwen3.8 Max | 221 | 38.0 | 59.3 | 1.8 | 0.0 | 0.9 |
| Gemini 3.7 Flash | 222 | 39.6 | 59.9 | 0.0 | 0.0 | 0.5 |
| Kimi K3 | 226 | 54.0 | 41.6 | 4.4 | 0.0 | 0.0 |
| Qwen3.8 Flash | 251 | 26.3 | 61.4 | 12.4 | 0.0 | 0.0 |

0 20 40 60 80 100

**F1** Declared DONE; scored zero
**F2** Reached the decision-turn limit
**F3** Explicitly declared FAIL
**F4** Exceeded runtime limit
**F5** Closed desktop unexpectedly

Figure 5: **Failure modes.** DONE/FAIL are agent-declared termination signals; F1 receives zero credit from the verifier. Missing-field and missing-file failures are excluded.

### 4.3 Ablation and Analysis

Table 3: **Ablation on initial-image presentation.** EngiScore with reference drawings in the environment (Env.) or the message (Msg.).

| | GUI | | CLI | |
|---|---|---|---|---|
| **Model** | Env. | Msg. | Env. | Msg. |
| GPT-5.6 Sol | **15.0** | **31.7** | **63.3** | **68.3** |
| Gemini 3.7 Flash | 3.3 | 10.0 | 38.3 | 40.0 |
| Kimi K3 | 6.7 | 10.0 | 40.0 | 36.7 |

Table 4: **Ablation on runtime image access in CLI.** EngiScore with `readimg` disabled or enabled.

| | w/ Init. Image | | w/o Init. Image | |
|---|---|---|---|---|
| **Model** | Off | On | Off | On |
| GPT-5.6 Sol | **66.7** | **68.3** | **48.1** | **41.5** |
| Gemini 3.7 Flash | 36.7 | 40.0 | 33.3 | 36.6 |
| Kimi K3 | 41.7 | 36.7 | 31.5 | 26.5 |

We study how observation design and interaction history affect agent performance. The ablations vary reference-image presentation, runtime image access, accessibility information, screenshot resolution, and history length for GPT-5.6 Sol, Gemini 3.7 Flash, and Kimi K3. Detailed task-level gains and regressions are provided in App. C.4.

**Initial-image presentation.** Presenting reference drawings directly in the task message improves GUI performance for all three models, with GPT increasing from 15.0 to 31.7 (Table 3). The benefit is smaller on the CLI interface, where agents can retrieve the drawings through `readimg`, and reverses for Kimi. The aggregate improvement therefore does not imply uniform gains: for GPT, five tasks that were previously successful become failures after the presentation format changes. This result suggests that easier access to visual inputs can help agents locate task-relevant information, but does not ensure that they use it correctly.

**Runtime image access.** Runtime access to images improves Gemini on both CLI conditions but reduces Kimi's score in both settings (Table 4). GPT benefits when an initial drawing is available and declines when it is not. Visual feedback can support artifact inspection and correction, but it can also trigger unnecessary revisions. The opposing effects across models indicate that the value of additional visual evidence depends on how an agent incorporates it into its execution strategy.

**Accessibility information.** Adding an accessibility tree improves Kimi, changes GPT little, and lowers Gemini's score (Figure 6a). Although all three models use fewer decision rounds, the additional structured information increases API cost by 50.7–147.9%. Accessibility metadata can therefore shorten interaction sequences without improving the resulting artifact, showing that more structured observations do not automatically translate into better engineering decisions.

**Screenshot resolution.** The native $1920 \times 1080$ resolution produces the highest score for all three models (Figure 6b). Reducing the resolution to $0.6\times$ lowers cost for GPT by 30.8% while preserving most of its score, but causes a much larger degradation for Kimi. At the task level, increasing resolution benefits Kimi on eleven tasks without losses, whereas GPT gains on ten and loses on ten.

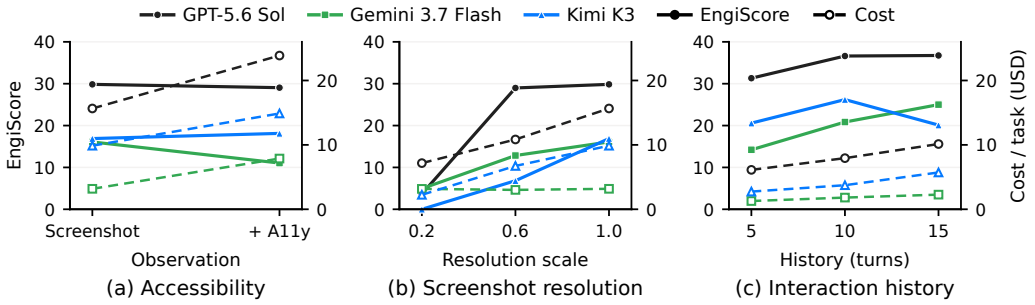



Figure 6: **Ablation on observation and history.** Accessibility and resolution use GUI tasks; history includes both interfaces. A11y adds an accessibility tree to the screenshot. Resolution scales are relative to $1920 \times 1080$.

Higher visual fidelity thus improves average performance, but its benefit depends on the model and does not apply uniformly across tasks.

**Interaction history.** The optimal history length is model dependent (Figure 6c). Gemini benefits from extending the window from five to fifteen turns, GPT largely plateaus after ten, and Kimi performs best at ten. For Kimi, extending the history from ten to fifteen turns lowers EngiScore while increasing cost by 53.0%.

### 4.4 Failure Analysis

We classify failures by their terminal outcome: `DONE` with a zero score, decision-turn exhaustion, explicit `FAIL`, runtime-limit violation, and unexpected desktop closure or restart. The first two categories account for 94.1% of failures in the main evaluation (Figure 5), indicating that most unsuccessful runs either terminate before satisfying the verifier or fail to complete within the available interaction budget. These two dominant failure modes correspond to distinct breakages of the design loop: declaring completion without verified artifacts reflects a failure at the validation stage, whereas decision-turn exhaustion reflects a breakdown during the construction stage.

**Completion declarations.** Declared completion is the dominant failure mode for GPT and Claude: 87.0% and 73.9% of their retained failures end with `DONE`, compared with 39.6% for Gemini. These episodes produce a terminal signal without satisfying the artifact checks, often because of incorrect geometry or broken consistency across workflow stages (Table D.3). The gap between declared completion and verified success highlights the difficulty of judging engineering correctness from the visible software state alone.

**Execution limits.** Decision-turn exhaustion is most common for Gemini, both Qwen models, and DeepSeek, and occurs more frequently on the GUI interface than on the CLI interface (Figure D.2). It accounts for 55.3% of GUI failures and 36.6% of CLI failures. Explicit `FAIL`, runtime-limit failures, and desktop interruptions are much less frequent, while runtime-limit failures are concentrated in a small number of long-running executions, including 28 DeepSeek runs and 9 Claude runs that reach the five-hour limit.

## 5 Conclusion

We introduced EngiWorld, a benchmark structured around the complete design loop, to evaluate LLM and MLLM agents on professional engineering workflows spanning 6 domains: CAD, CAE, CAM, BIM, EDA, and 3D visualization. The benchmark contains 1,301 expert-curated tasks across 26 software platforms and 6 task types, with GUI and CLI interfaces covering individual tools and multiple-stage workflows. Its artifact-centric evaluation, built on a unified domain-verifier suite, checks submitted artifacts for geometric, physical, connectivity, simulation, and cross-stage consistency, while preserving continuous quality scores for quantitative designs. Across seven frontier agents, the strongest reaches an EngiScore of 44.3, but only 3.6% of multi-software attempts succeed, with failures concentrated in unsatisfied engineering constraints and deliverables that agents did not verify before completion. This result exposes a fundamental gap between tool operation and reliable engineering execution, positioning EngiWorld as a rigorous foundation for building industrial agents that go beyond operating individual tools to close the engineering design loop.

### AI USE STATEMENT

Generative AI assisted task authoring and manuscript preparation, including drafting, language editing, and reference lookup. For task authoring, models proposed candidate specifications and evaluator components under an expert-calibrated process; domain experts reviewed the outputs and validated the final tasks as described in Section 3.3 and App. A. The evaluated models generated engineering artifacts as experimental outputs, which were assessed by task-specific verifiers. The authors take responsibility for the manuscript, benchmark materials, and reported findings.

### ETHICS STATEMENT

ENGIWORLD evaluates engineering agents in virtual software environments. Passing its artifact checks establishes compliance with specified task requirements, not certification for manufacturing, construction, or other safety-critical deployment. Practical use of generated designs requires independent engineering review and applicable safety validation. Tasks adapted from software documentation, tutorials, and community resources require attention to attribution and redistribution rights. Access to proprietary applications and redistribution of software images or third-party assets remain subject to their respective licenses. Benchmark performance should be interpreted within the evaluated software, tasks, and verifier coverage.

### REPRODUCIBILITY STATEMENT

Section 4.1 specifies the evaluated models, task-subset allocation, interaction budgets, and metrics. App. B.1 describes application versions, virtual-machine initialization, and runtime dependencies; App. B.2 defines artifact checks, tolerances, and quantitative scoring. The verifier audit and validation procedures are documented in App. B.3. The accompanying release will provide the task manifest, the 300-task selection list, the archived prompt specification, and the data and scripts used to generate the reported figures. Reproducing agent executions additionally requires the task assets, execution harness, verifiers, and access to the specified software environments.

## REFERENCES

Alibaba Cloud. Model Studio: Supported Models and Capabilities Overview, 2026. URL https://www.alibabacloud.com/help/en/model-studio/models. Accessed September 24, 2026.

Anthropic. Introducing Claude Opus 5, 2026. URL https://www.anthropic.com/news/claude-opus-5. Accessed September 24, 2026.

Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Yuchen Mao, Wenjing Hu, Tianbao Xie, Hongshen Xu, Danyang Zhang, Sida Wang, Ruoxi Sun, Pengcheng Yin, Caiming Xiong, Ansong Ni, Qian Liu, Victor Zhong, Lu Chen, Kai Yu, and Tao Yu. Spider2-V: How Far Are Multimodal Agents From Automating Data Science and Engineering Workflows? In *Advances in Neural Information Processing Systems*, volume 37, pp. 107703–107744, 2024. doi: 10.52202/079017-3421. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c2f71567cd53464161cab3336e8fc865-Abstract-Datasets_and_Benchmarks_Track.html.

Yuxuan Chen, Xu Zhu, Hua Zhou, and Zhuyin Ren. MetaOpenFOAM: an LLM-based multi-agent framework for CFD. arXiv preprint arXiv:2407.21320, 2024. URL https://arxiv.org/abs/2407.21320.

DeepSeek. DeepSeek API Documentation, 2026. URL https://api-docs.deepseek.com/. Accessed September 24, 2026.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam 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. doi: 10.52202/075280-1220. URL https://proceedings.neurips.cc/paper_files/paper/2023/

hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Zihan Dong, Yuanzhe Liu, Zhiyuan Ma, Qishi Zhan, Dehan Kong, Guohao Li, and Kaixin Li. CAD-World: Computer-Use Benchmark for Long-Horizon Computer-Aided Design. arXiv preprint arXiv:2609.16251, 2026. URL https://arxiv.org/abs/2609.16251.

Maggie Y. Gao, Chao Li, Frank Petzold, Robert L.K. Tiong, and Yaowen Yang. Lifecycle framework for AI-driven parametric generative design in industrialized construction. *Automation in Construction*, 174:106146, 2025. doi: 10.1016/j.autcon.2025.106146. URL https://doi.org/10.1016/j.autcon.2025.106146.

Yifei Gong, Xing Wu, Wenda Liu, and Tukang. TOOLCAD: Exploring Tool-Using Large Language Models in Text-to-CAD Generation with Reinforcement Learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23161–23188, 2026. doi: 10.18653/v1/2026.findings-acl.1160. URL https://aclanthology.org/2026.findings-acl.1160/.

Google DeepMind. Gemini 3.7 Flash, 2026. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.7-flash. Accessed September 24, 2026.

Yunqi Gu, Ian Huang, Jihyeon Je, Guandao Yang, and Leonidas Guibas. BlenderGym: Benchmarking Foundational Model Systems for Graphics Editing. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18574–18583, 2025. doi: 10.1109/cvpr52734.2025.01731. URL https://doi.org/10.1109/cvpr52734.2025.01731.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation. In *International Conference on Machine Learning*, volume 235, pp. 20271–20309, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D.D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. UV-Net: Learning From Boundary Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11703–11712, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Jayaraman_UV-Net_Learning_From_Boundary_Representations_CVPR_2021_paper.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? arXiv preprint arXiv:2310.06770, 2023. URL https://arxiv.org/abs/2310.06770.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998. doi: 10.1016/s0004-3702(98)00023-x. URL https://doi.org/10.1016/s0004-3702(98)00023-x.

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 *European Conference on Computer Vision*, pp. 161–178, 2024. doi: 10.1007/978-3-031-73113-6_10. URL https://doi.org/10.1007/978-3-031-73113-6_10.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2CAD: Generating Sequential CAD Designs from Beginner-to-Expert Level Text Prompts. In *Advances in Neural Information Processing Systems*, pp. 7552–7579, 2024. doi: 10.52202/079017-0242. URL https://doi.org/10.52202/079017-0242.

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 (Volume 1: Long Papers)*, pp. 881–905, 2024. doi: 10.18653/v1/2024.acl-long.50. URL https://doi.org/10.18653/v1/2024.acl-long.50.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation. In *International Conference on Machine Learning*, pp. 18319–18345, 2023. URL https://proceedings.mlr.press/v202/lai23b.html.

Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Free2CAD: parsing freehand drawings into CAD commands. *ACM Transactions on Graphics*, 41(4):1–16, 2022. doi: 10.1145/3528223.3530133. URL https://doi.org/10.1145/3528223.3530133.

Chunyi Li, Longfei Li, Zicheng Zhang, Xiaohong Liu, Min Tang, Weisi Lin, and Guangtao Zhai. Using GUI Agent for Electronic Design Automation. arXiv preprint arXiv:2512.11611, 2025. URL https://arxiv.org/abs/2512.11611.

Mingjie Liu, Nathaniel Pinckney, Brucek Khailany, and Haoxing Ren. Invited Paper: VerilogEval: Evaluating Large Language Models for Verilog Code Generation. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, 2023. doi: 10.1109/iccad57390.2023.10323812. URL https://doi.org/10.1109/iccad57390.2023.10323812.

Shang Liu, Wenji Fang, Yao Lu, Qijun Zhang, Hongce Zhang, and Zhiyao Xie. RTLCoder: Outperforming GPT-3.5 in Design RTL Generation with Our Open-Source Dataset and Lightweight Solution. In *2024 IEEE LLM Aided Design Workshop (LAD)*, pp. 1–5, 2024. doi: 10.1109/lad62341.2024.10691788. URL https://doi.org/10.1109/lad62341.2024.10691788.

Songkai Liu, Yanqing Shen, Yilun Zhang, Zhangli Hou, Xin Wang, Jianxi Luo, and Zhinan Zhang. iDesignGPT enhances conceptual design via large language model agentic workflows. *Nature Communications*, 17(1):1997, 2026. doi: 10.1038/s41467-026-68672-1. URL https://doi.org/10.1038/s41467-026-68672-1.

Yao Lu, Shang Liu, Qijun Zhang, and Zhiyao Xie. RTLLM: An Open-Source Benchmark for Design RTL Generation with Large Language Model. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 722–727, 2024. doi: 10.1109/asp-dac58780.2024.10473904. URL https://doi.org/10.1109/asp-dac58780.2024.10473904.

Dimitrios Mallis, Marco Wang, Ahmet Serdar Karadeniz, Elisa Ricci, Anis Kacem, and Djamila Aouada. Text-to-CAD Evaluation with CADTests. arXiv preprint arXiv:2605.07807, 2026. URL https://arxiv.org/abs/2605.07807.

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, 2025. doi: 10.52202/085713-1217. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/33b0a2df370f0152e74a4f088ed36ccf-Abstract-Datasets_and_Benchmarks_Track.html.

Moonshot AI. Kimi K3 Tech Blog: Open Frontier Intelligence, 2026. URL https://www.kimi.com/en/blog/kimi-k3. Accessed September 24, 2026.

Nayantara Mudur, Hao Cui, Subhashini Venugopalan, Paul Raccuglia, Michael P. Brenner, and Peter Norgaard. FEABench: Evaluating Language Models on Multiphysics Reasoning Ability. arXiv preprint arXiv:2504.06260, 2025. URL https://arxiv.org/abs/2504.06260.

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. URL https://arxiv.org/abs/2606.20146.

OpenAI. GPT-5.6 Sol Model, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol. Accessed September 24, 2026.

Christopher 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. arXiv preprint arXiv:2405.14573, 2024. URL https://arxiv.org/abs/2405.14573.

Lei Ren, Haiteng Wang, Jiabao Dong, Zidi Jia, Shixiang Li, Yuqing Wang, Yuanjun Laili, Di Huang, Lin Zhang, and Bohu Li. Industrial Foundation Model. *IEEE Transactions on Cybernetics*, 55(5): 2286–2301, 2025. doi: 10.1109/tcyb.2025.3527632. URL https://doi.org/10.1109/tcyb.2025.3527632.

Harmanjot Singh, Abhra Dubey, and Jorge Alejandro Amador Herrera. CADEngBench: It Looks Like CAD, but Does It Work? Evaluating Parametric Design, Assembly Reasoning, and Physics Simulation. arXiv preprint arXiv:2608.09296, 2026. URL https://arxiv.org/abs/2608.09296.

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*, 2026a. URL https://arxiv.org/abs/2505.19897.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, Haoyun Zhang, Jeffrey Lin, Weishu Zhang, Tyler Zeng, Ying Yan, Bo Liu, Hanson Wen, Mingyang Xu, Xiaoyuan Liu, Zimeng Chen, Weiyan Shi, Amanda Dsouza, Vincent Sunn Chen, Patrick Bryant, Carl Boettiger, Yamini Rangan, Bradley Rothenberg, Kyle Steinfeld, Arvind Rao, Tapio Schneider, Georgios Yannakakis, Laure Zanna, Kaan Ozbay, Ida Sim, Tarek Zohdi, George Em Karniadakis, Jack Gallant, Teresa Head-Gordon, Yushan Li, Wenxi Deng, Tao Sun, Huiqi Wang, Zhun Wang, Justin Xu, Chris Yuhao Liu, Yafei Cheng, Rongwang Hu, Aras Bacho, Shengcao Cao, Zengyi Qin, Yixiong Chen, Hengduan Fan, Hao Liu, Lin Zeng, Shashank Muralidhar Bharadwaj, Litian Gong, Yingxuan Yang, Maojia Song, Ruheng Wang, Zongzheng Zhang, Honglin Bao, Shuo Lu, Jianhong Tu, Zhonghua Wang, Zheng Zhang, Zijiao Chen, Yanqiong Jiang, Zhendong Li, Bohan Lyu, Chang Ma, Peiran Xu, Benran Zhang, Shangding Gu, Haoyue Hua, Haoyang Li, Wanzhe Liao, Chengzhi Liu, Junbo Peng, Haoran Sun, Zechen Xu, Bo Chen, Jiayi Cheng, Yi Jiang, Keying Kuang, Yuan Li, Youbang Pan, Ziyan Rao, Alexander Schubert, Yifan Shen, Vincent Siu, Xiatao Sun, Kangqi Zhang, Xiaopan Zhang, Yuchen Zhu, Ishaan Singh Chandok, Lei Ding, Jingxuan Fan, Andrew Glover, Jiaming Hu, Yiran Hu, Wenbo Huang, Zixin Jiang, Haoran Jin, Lukas Kim, Ming Liu, Yang Liu, Alireza Rafiei, Xuhuan Shen, Kunyang Sun, Sophia Sun, Ting Sun, Eric Wang, Yixin Wang, Hanwen Xing, Sihan Xu, Yuzheng Xu, Zhongxing Xu, Zhiling Yan, Boqin Yuan, Ruiqi Zhang, Yifan Zhang, Zibo Zhao, Liana, Santanu Bosu Antu, Haoyue Bai, Carlo Bosio, Joseph Cavanagh, Patricia Cavazos-Rehg, Tianxing Chen, Xuewen Chen, Yipu Chen, Chenyu Zhu, Chen Dai, Stefano De Castro, Yunfu Deng, Kaustubh Dhole, Jiayuan Ding, Chenchen Du, Zhehang Du, Hao Fan, Run-Ze Fan, Hengyu Fu, Shi Gu, Yifan Gu, Charlie Guo, Baihe Huang, Baixiang Huang, Rimika Jaiswal, Zhihan Jiang, Ran Jin, Erin Kasson, Xin Lan, Joseph Lee, Deren Lei, Chenyu Li, Daofeng Li, Haitao Li, Hongwei Li, Jingyan Li, Xiao Li, Yi Li, Yinsheng Li, Yuangang Li, Zhixu Li, Wenyu Liang, Longtai Liao, Kevin Qinghong Lin, Andy Zeyi Liu, Che Liu, Jiaming Liu, Kaiyuan Liu, Xuan Liu, Pan Lu, Wenbo Lv, Yicheng Lyu, Qiuyang Mang, Kyle Montgomery, Yuzhou Nie, Ruoxi Ning, Jorin Overwiening, Xu Pan, Layna Paraboschi, Core Francisco Park, Justin Purnomo, Swati Rajwal, Scott Rankin, Bixuan Ren, Yiren Rong, HaoYang Shang, Ventus Shaw, Fiona Shen, Jiawei Shen, Minqi Shi, Shi Qiu, Huaxiu Yao, Tianneng Shi, Jonah So, Vladislav Susoy, Hannah Szlyk, Haocheng Wang, Jialu Wang, Wei Wang, Xinyu Wang, Zehao Wang, Dowling Wong, Angela Wu, Dehao Wu, Fangyu Wu, Mengyuan "Millie" Wu, Yu Wu, Yuchen Wu, Yuhao Wu, Qingpo Wuwu, Weihang Xiao, Yongyi Xiong, Fan Xu, Ruiling Xu, Mingxuan Yan, Benjamin Yang, Jirong Yang, Sen Yang, Xiaoli Yang, Yushi Yang, Haoran Ye, Xiaohu Yu, Zhengming Yu, Chenlong Zhang, Chi Zhang, Hanning Zhang, Hanwen Zhang, Junge Zhang, Kunpeng Zhang, Song Zhang, Wenjin Zhang, Wenshuo Zhang, Ying Zhang, Yizhi Zhang, Brian Zhao, Qijian Zhao, Yimin Zhao, Yuhaohua Zheng, Liwei Zhou, Tianyue Zhou, Sichen Zhu, Siqi Zhu, Yan Zhu, Yishu Zhu, Jierui Zuo, Chonghao Cai, Helena Casademunt, Wenjia Chen, Cheng Cheng, Nawen Deng, Rao Fu, Tianfu Fu, Yifan Han, He Ren, Zhenyu He, Qiao Jin, Langlang Li, Yuetai Li, Sylvia Liu, Lu Lu, Luqing Zhou, Subhabrata Mukherjee, Yunqi Ouyang, Yin Ren, Dawei Shi, Haoran Wu, Zhiyue Wu, Hannah Yao, Zhuoran Yi, Jenny Yu, Rhea Zhan, Hang Zhou, Blake Zhu, Junfan Zhu, Alan Yuille, Yang Liu, Russell Alan Poldrack, Jiachen Li, Zhenglu Li, Molei Tao, Jing Huang, Wenqi Shi, Costas Spanos, Lichao Sun, Chenguang Wang, Orson Xu, Zhen Dong, Hector Gomez, Aylin Caliskan, Ali Emami, Haimin Hu, Zhi Li, Lihui Liu, Murphy Niu,

Yi Shao, Jianxin Sun, Mikko Tolonen, Ting Wang, Sanjiv Das, Yanjun Gao, Wenbo Guo, Erika J Schneider, Zhiyong Lu, Yian Ma, Mark Mueller, Radha Poovendran, Somayeh Sojoudi, Yinglun Zhu, and Dawn Song. Agents' Last Exam. arXiv preprint arXiv:2606.05405, 2026b. URL https://arxiv.org/abs/2606.05405.

Haoyuan Wu, Zhuolun He, Xinyun Zhang, Xufeng Yao, Su Zheng, Haisheng Zheng, and Bei Yu. ChatEDA: A Large Language Model Powered Autonomous Agent for EDA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(10):3184–3197, 2024. doi: 10.1109/tcad.2024.3383347. URL https://doi.org/10.1109/tcad.2024.3383347.

Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6772–6782, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Wu_DeepCAD_A_Deep_Generative_Network_for_Computer-Aided_Design_Models_ICCV_2021_paper.html.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J. Hua, 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. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 23826–23854, 2023. doi: 10.52202/075280-1035. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4b175d846fb008d540d233c188379ff9-Abstract-Datasets_and_Benchmarks.html.

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. URL https://arxiv.org/abs/2606.29537.

Ling Yue, Nithin Somasekharan, Tingwen Zhang, Yadi Cao, and Shaowu Pan. Foam-Agent 2.0: An End-to-End Composable Multi-Agent Framework for Automating CFD Simulation in OpenFOAM. arXiv preprint arXiv:2509.18178, 2025. URL https://arxiv.org/abs/2509.18178.

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. arXiv preprint arXiv:2307.13854, 2023. URL https://arxiv.org/abs/2307.13854.

# Supplementary Material

## A Dataset Construction

EngiWorld is constructed through an expert-led, AI-assisted pipeline grounded in real engineering workflows and professional software environments. As illustrated in Fig. A.1, domain experts define the task objectives, engineering artifacts, and evaluation criteria, while AI is used to standardize task specifications, assist artifact construction, and implement programmatic verifiers. Each stage is reviewed and iteratively refined by experts before the resulting task is included in the benchmark. The construction process consists of three stages: task construction, task-instance construction, and evaluation-criteria construction.

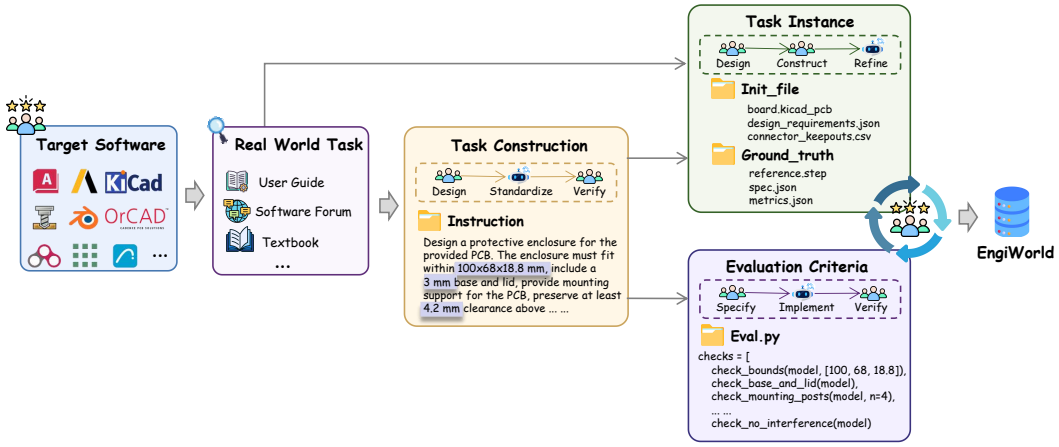



Figure A.1: **Dataset construction pipeline.** Representative engineering scenarios are transformed into executable benchmark tasks through task construction, task-instance construction, and evaluation-criteria construction, with expert review and validation throughout the process.

### A.1 Task Construction

Task construction begins with the selection of software platforms and workbenches across the six engineering domains covered by EngiWorld. For each environment, experts identify representative scenarios from real engineering workflows and software-specific resources, including official documentation, user guides, tutorials, technical references, and engineering textbooks. These scenarios are selected to reflect the characteristic capabilities and workflows of the corresponding software.

Each scenario is formulated as a result-oriented task specification that defines the engineering objective, available inputs, required deliverables, and relevant constraints. AI assists in standardizing the structure and wording of the specification so that tasks across heterogeneous software environments follow a consistent format. For tasks initialized from existing engineering artifacts, observable properties remain encoded in the provided task state, allowing the instruction to focus on the intended outcome and required deliverables. Experts then review and finalize the specification, which serves as the common basis for task instantiation and evaluation.

### A.2 Task Instance Construction

Once the task specification is finalized, the corresponding task instance is constructed in the designated software environment. Experts prepare the initial engineering state and construct a validated reference realization in the corresponding professional software, with AI assistance during artifact preparation and iterative refinement. Depending on the domain, a task instance may involve CAD geometry, PCB designs, simulation models, BIM models, manufacturing files, structured parameters, or 3D scenes.

The reference realization provides a verified solution and supports the extraction of task-specific engineering quantities used during construction and validation. Multiple valid realizations can satisfy the same task specification when they meet the required engineering conditions. For multi-software tasks, intermediate artifacts preserve the dependencies between successive workflow stages and may serve as inputs to subsequent applications. For open-choice tasks, the engineering requirements and deliverables are fixed while the agent retains flexibility in selecting tools and execution order.

### A.3 Evaluation Criteria and Verification

Evaluation criteria are derived directly from the engineering requirements of each task. Experts specify the properties and tolerances that characterize successful completion, covering quantities such as geometry and topology, component placement, clearances, connectivity, physical or simulation outputs, manufacturing constraints, and cross-artifact relationships. Together, these criteria define the engineering conditions under which a submitted artifact is considered valid.

AI assists in translating these expert-defined criteria into a task-specific programmatic verifier. The verifier extracts the relevant properties from the submitted final and, when required, intermediate artifacts and applies the corresponding checks. For example, a CAD task may verify dimensions, material regions, mounting structures, openings, clearances, and interference directly from the submitted geometry, while a simulation task may evaluate solver outputs and derived physical quantities. Numerical requirements are evaluated using explicit task-specific tolerances.

The verifier is validated together with the task instance in the target software environment. Reference realizations are used to confirm the expected acceptance behavior, while controlled variants are used to exercise task-relevant failure conditions and numerical boundaries. For tasks admitting multiple valid realizations, alternative solutions are additionally examined when applicable. Experts finally review the instruction, task instance, reference realization, and verifier as a complete task package. Tasks with identified inconsistencies are revised and revalidated before inclusion in EngiWorld.

## B Engineering Evaluation

### B.1 Software Environments

Each task runs in an isolated Windows or Ubuntu virtual machine with the required engineering software installed. EngiWorld provides 37 environments: 26 software-specific environments, 10 composite environments supporting multi-software workflows, and one dedicated environment for Open-ended tasks. Table B.1 lists the 26 software-specific environments, with FreeCAD Modeling and Path treated as separate workbench environments. Before each episode, the harness restores a clean snapshot, copies the task inputs, prepares the workspace, and opens the relevant documents. Cross-application tasks use a shared workspace containing the participating applications. The software versions listed in Table B.1 refer to the software-specific environments; composite environments may use different versions. For example, the Archicad–OpenStudio workflow uses OpenStudio 3.10 on Windows.

After an episode ends, the verifier runs inside the task VM and uses the application's data model or a format-specific parser. The checker stack includes CadQuery/OCCT for solids, ezdxf for drawings, IfcOpenShell for building models, native FreeCAD and Blender APIs, Abaqus `odbAccess` for simulation results, and SQLite for EnergyPlus/OpenStudio outputs. Observation, budget, and history settings follow Sec. 4.1.

### B.2 Artifact Checks and Scoring

**Artifact inspection.** After termination, the verifier reopens the submitted deliverables and extracts the properties required by the task. Neutral formats such as STEP, DXF, and IFC are re-imported so that export errors affect the result. Numerical properties use task-specific tolerances, while connectivity, entity roles, and required references use exact checks (Table B.2). Missing or unreadable deliverables fail the corresponding criteria. When a file format permits equivalent names or representations, the checker accepts them while retaining the required geometry, physical settings, and numerical constraints. Integrity checks protect designated inputs and prevent scripted bypasses of

Table B.1: **Software environments.** The 37 environment configurations in EngiWorld: 26 software-specific environments, 10 composite environments, and one Open-ended environment.

| Domain / type | Application(s) | Version | OS | Interface |
|---|---|---|---|---|
| **Software-specific environments (26)** | | | | |
| CAD | FreeCAD | 0.21.2 | Ubuntu | CLI/GUI |
| | AutoCAD | 2024 | Windows | CLI/GUI |
| | SolidWorks | 2025 | Windows | CLI/GUI |
| | BRL-CAD | 7.32.2 | Ubuntu | CLI |
| | OpenSCAD | 2021.01 | Ubuntu | CLI/GUI |
| | LibreCAD | 2.2.0.2 | Ubuntu | GUI |
| | SolveSpace | 3.1 | Ubuntu | GUI |
| CAE | ANSYS | 2026R | Windows | CLI/GUI |
| | Abaqus | 2025L | Windows | CLI/GUI |
| | OpenFOAM | 11 | Ubuntu | CLI |
| | FEniCS (DOLFIN) | 2019.1.0 | Ubuntu | CLI |
| | CalculiX | 2.21 | Ubuntu | CLI |
| | FLORIS | 4.6.4 | Ubuntu | CLI |
| | OpenFAST | 5 | Ubuntu | CLI |
| EDA | Cadence OrCAD | 24.1 | Windows | CLI/GUI |
| | EAGLE | 7.7.0 | Ubuntu | CLI/GUI |
| | Altium Designer | 17 | Windows | CLI/GUI |
| | KiCad | 10.0.2 | Ubuntu | CLI/GUI |
| CAM | FreeCAD Path | 0.21.2 | Ubuntu | CLI/GUI |
| | SolidCAM | 2025 | Windows | CLI/GUI |
| BIM | Archicad | 27 | Windows | CLI/GUI |
| | Bonsai | 0.8.5 | Ubuntu | CLI/GUI |
| | OpenStudio | 1.11.0 | Ubuntu | CLI/GUI |
| | Revit | 2025 | Windows | CLI/GUI |
| 3D/Vis. | Blender | 4.2.3 | Ubuntu | CLI/GUI |
| | ZBrush | 2025 | Windows | GUI |
| **Composite environments: Multi-software tasks (6)** | | | | |
| Multi | AutoCAD + SolidWorks + Abaqus | — | Windows | GUI |
| | SolidWorks + Abaqus | — | Windows | GUI |
| | Archicad + OpenStudio | — | Windows | CLI |
| | Revit + Archicad + OpenStudio | — | Windows | CLI |
| | LibreCAD + FreeCAD + KiCad + Blender | — | Ubuntu | GUI |
| | KiCad + OpenSCAD + FreeCAD + Blender | — | Ubuntu | CLI |
| **Composite environments: Software-selection tasks (4)** | | | | |
| Selection | ANSYS + Abaqus + AutoCAD | — | Windows | CLI/GUI |
| | OrCAD + Altium Designer + OpenSCAD | — | Windows | CLI/GUI |
| | Revit + Archicad + Abaqus | — | Windows | CLI/GUI |
| | FreeCAD + LibreCAD + KiCad | — | Ubuntu | GUI |
| **Open-ended environment (1)** | | | | |
| Open-ended | Dedicated workspace with freely chosen tools | — | Ubuntu | CLI |

the GUI interface; implementation-specific signing and certificate rules are described by the corresponding environment configuration.

**Cross-stage consistency.** Multi-software verifiers check intermediate and final artifacts against the task inputs. In a KiCad–OpenSCAD–FreeCAD–Blender workflow, the checks match board identifiers and mounting-hole dimensions to the mechanical map, then verify that the enclosure, assembly, and review scene preserve those features. A missing upstream artifact fails every dependent criterion.

**Quantitative scoring.** Quantitative tasks specify an objective $m$ and hard feasibility constraints $F_i$ (Equation 2), including required geometry, material and load assignments, and task-specific limits on physical response. Each task defines a quality function for feasible outputs. When the objective measures improvement over an initial design, the score is normalized between a baseline $m_{\text{base}}$ and an improved reference $m_{\text{ref}}$:

$$Q_i(\mathcal{Y}_T) = \begin{cases} \text{clip}_{[0,1]}\left(\dfrac{m(\mathcal{Y}_T) - m_{\text{base}}}{m_{\text{ref}} - m_{\text{base}}}\right), & F_i(\mathcal{Y}_T) = 1, \\ 0, & F_i(\mathcal{Y}_T) = 0. \end{cases} \tag{4}$$

The normalization supports both minimization and maximization objectives: matching the baseline gives zero, and matching or exceeding the reference gives one. Other tasks score design quality directly, such as layout cost or surface approximation error, so a positive score need not indicate improvement over the initial artifact. Infeasible outputs receive zero regardless of the objective value. Raw measurements accompany $Q_i$ in App. C.2; aggregate performance uses EngiScore (Equation 3).

### B.3 Verifier Validation

Expected outcomes are assigned from the task requirements before verifier execution. Task authors label alternative and defective artifacts, and a second annotator reviews these judgments independently of the verifier output. Disagreements trigger a review of the specification and a repeat of the affected checks.

All 96 reference artifacts in the completed audit group pass after review and repair. The alternative-solution tests cover the Abaqus/ANSYS Software-selection family and FreeCAD lightweight design: all valid submissions are accepted, while empty and metrics-only submissions are rejected (Table B.3). Examples of artifact violations in agent-declared successes appear in App. D.2; tolerance-boundary tests are outside the completed audit populations.

## C Additional Results and Analyses

Model names in this appendix omit inference settings; all runs use the configurations specified in Sec. 4.1.

### C.1 Tasks Missed by All Models

Across the 300 main-evaluation tasks, 128 receive a zero score from all seven models (Fig. C.1). Among the remaining 172 tasks, 37 are solved by one model, 35 by two, and 29 by all seven. The all-zero set is concentrated in Single-software tasks, but also includes 22 Multi-software, 18 Software-selection, and 12 Quantitative design tasks. This distribution shows that the aggregate performance gap is not caused only by a small number of difficult cross-application workflows; many individual application tasks remain unsolved by every evaluated model.

### C.2 Quantitative Design Results

Partial scores occur in 53 of the 231 main-evaluation runs on Quantitative design tasks; nine runs receive 1.0 and 169 receive zero (Table C.1). These runs cover 33 tasks and seven models. KiCad layout and ZBrush surface tasks account for 38 of the 53 partial scores. Task scores distinguish

Table B.2: **Artifact checks.** Representative properties and acceptance rules across the verifier families. Numerical tolerances are task specific.

| Check family | Properties | Acceptance criteria |
|---|---|---|
| Global geometry | Bounding spans, volume, surface area, and solid validity | Dimensional bounds and topology checks |
| Local geometry | Profiles at specified sections in the task's datum frame | Profile tolerances, e.g., 0.005–0.08 mm |
| Solid–void occupancy | Material, cavities, through-holes, and bore clearance | Required occupancy; residual material, e.g., $\leq 0.1\,\mathrm{mm}^3$ |
| Engineering assignments | Materials, loads, boundary conditions, and thermal zones | Required assignments and parameter values |
| Simulation outputs | Field quantities and physical responses | Task-specific absolute or relative error bounds |
| Structural relations | Net connectivity, entity roles, and cross-file references | Exact connectivity and semantic consistency |

Table B.3: **Verifier validation.** Acceptance outcomes for reference, alternative-valid, and defective artifact populations.

| Test population | Scope | $n$ | Accepted | Rejected |
|---|---|---|---|---|
| Reference artifacts | Completed audit group | 96 | 96 | 0 |
| Alternative solver | Abaqus / ANSYS, CLI | 40 | 40 | 0 |
| Alternative solver | Abaqus / ANSYS, GUI | 40 | 40 | 0 |
| Alternative design | FreeCAD lightweight design | 5 | 5 | 0 |
| Empty delivery | Software-selection tasks, CLI / GUI | 21 | 0 | 21 |
| Metrics without artifacts | Software-selection tasks, CLI / GUI | 21 | 0 | 21 |

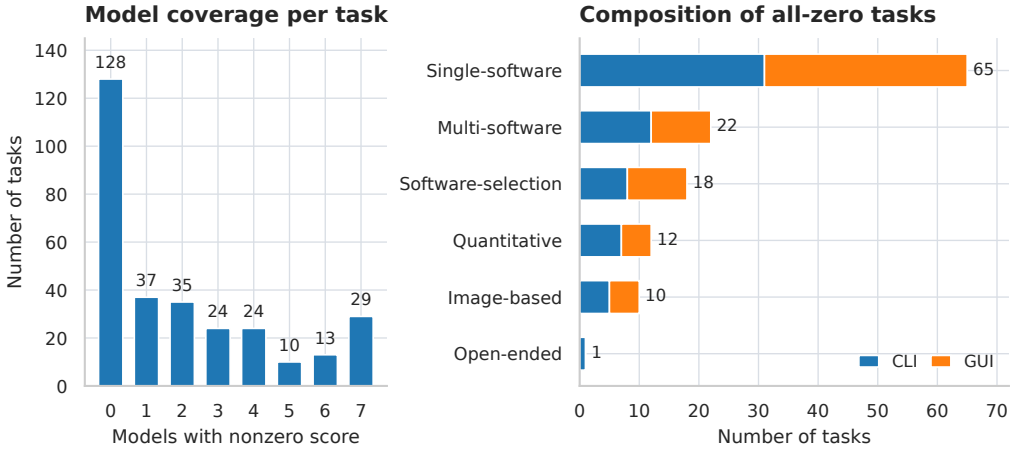



Figure C.1: **Tasks missed by all models.** Of 300 main-evaluation tasks, 128 receive zero from all seven models. The remaining bars show how many models obtain a nonzero score, while the composition bars partition the 128 all-zero tasks by type and interface.

design quality among valid outputs, while validity checks determine whether an artifact can receive credit. Objectives, checks, and scoring definitions follow App. B.2.

**Quality differences among valid outputs.** On ZBrush task 04, both Gemini 3.7 Flash and GPT-5.6 Sol produce valid surfaces with full evaluation-grid coverage. GPT achieves a lower height-field RMSE, 0.07650 versus 0.15098 OBJ units, and a higher score, 0.892 versus 0.804 (Table C.2). The initial design scores 0.880: GPT improves it slightly, whereas Gemini produces a valid surface

Table C.1: **Quantitative design outcomes.** Each model is evaluated on 33 tasks. Partial and full denote $0 < s < 1$ and $s = 1$, respectively; EngiScore includes zero-scored runs.

| | Number of runs | | | |
|---|---|---|---|---|
| **Model** | **Zero** | **Partial** | **Full** | **EngiScore** |
| **GPT-5.6 Sol** | **20** | **8** | **5** | **36.4** |
| Claude Opus 5 | 21 | 8 | 4 | 32.8 |
| Gemini 3.7 Flash | 24 | 9 | 0 | 20.6 |
| Kimi K3 | 24 | 9 | 0 | 21.9 |
| Qwen 3.8 Max | 24 | 9 | 0 | 22.3 |
| Qwen 3.8 Flash | 29 | 4 | 0 | 9.9 |
| DeepSeek V4.1 Flash | 27 | 6 | 0 | 16.2 |
| All runs | 169 | 53 | 9 | 22.9 |

of lower quality. For this task, the score measures the submitted surface's quality rather than its improvement over the initial design.

**Interpreting a near-full score.** Gemini's KiCad task-01 output scores 0.982, compared with 0.647 for the baseline layout. The evaluator-computed weighted cost decreases from 64.2085 to 3.2135, and route length decreases from 53.1618 to 40.1688 mm. All four movable components reach their target positions, with zero vias and minimum spacing of 4.6174 mm against a 2.0 mm requirement. The remaining cost comes from the route-length term (Table C.2). The gap to 1.0 therefore reflects residual engineering cost, not the fraction of requirements left unmet.

**Validity gates and zero scores.** On the same ZBrush task 04, Qwen 3.8 Flash obtains an RMSE of 0.18097 OBJ units but scores zero: its mesh contains degenerate faces, overlapping projected triangles, and multiple surface heights at the sampling points. GPT also scores zero on ZBrush task 05 because its grid coverage is 94.75%, below the required 98%. The RMSE remains a diagnostic measurement for these rejected artifacts, but it cannot compensate for invalid topology or insufficient coverage. Partial scores distinguish the quality of admissible designs; the validity gate prevents a geometrically defective output from receiving credit for approximation accuracy alone.

### C.3 Open-ended versus Multi-software Tasks

**Shared task cases**

**Task 02 Buck-regulator heat-spreader clamp package**

| | |
|---|---|
| **Task type** | Multi-software and Open-ended |
| **Shared design** | PA12 enclosure for a buck-regulator PCB with thermal contact between U1 and the lid. |
| **Geometry** | $117.2 \times 75.2 \times 18.6$ mm package; base, side walls, lid, and four bored PCB standoffs; $6 \times 6$ mm contact column. |
| **Hard constraints** | Contact column within 0.1 mm of U1; empty L1 exclusion region; at least 2.4 mm clearance above C1; side-wall openings for J1 and J2. |

**Task 09 Thermal-camera calibration-target PCB mount**

| | |
|---|---|
| **Task type** | Multi-software and Open-ended |
| **Shared design** | Black-PA12 mount for a calibration-target PCB with two exposed heaters. |
| **Geometry** | $131 \times 105 \times 25.1$ mm assembly; tray, separate lid, and four bored PCB standoffs; two independent 36 mm openings above H1 and H2. |
| **Hard constraints** | Preserve PCB placement and connector clearances; keep J1 and J2 accessible; each heater opening at least 99% open with an edge gap of at least 10 mm. |

Figure C.2: **Paired Multi-software and Open-ended tasks.** Tasks 02 and 09 share engineering targets but differ in workflow and delivery requirements.

Table C.2: **Representative quantitative outcomes.** Task-specific quality metrics are reported together with artifact validity, baseline quality, and EngiScore.

| **KiCad \| task 01: Layout optimization** | | | | | | |
|---|---|---|---|---|---|---|
| Model | Cost (USD) | Route (mm) | Spacing (mm) | Vias | Baseline | Score |
| Gemini 3.7 Flash | 3.2135 | 40.1688 | 4.6174 | 0 | 0.647 | 0.982 |
| **ZBrush \| task 04: Thumb-rest surface** | | | | | | |
| Model | RMSE | Coverage (%) | Faces | Valid | Baseline | Score |
| Gemini 3.7 Flash | 0.15098 | 100.00 | 256 | Pass | 0.880 | 0.804 |
| GPT-5.6 Sol | 0.07650 | 100.00 | 256 | Pass | 0.880 | 0.892 |
| Qwen 3.8 Flash | 0.18097 | 100.00 | 24,576 | Fail | 0.880 | 0.000 |
| **ZBrush \| task 05: Procedural ridge surface** | | | | | | |
| Model | RMSE | Coverage (%) | Faces | Valid | Baseline | Score |
| GPT-5.6 Sol | 0.49270 | 94.75 | 256 | Fail | 0.894 | 0.000 |

The main evaluation includes four Open-ended tasks, numbered 02, 07, 09, and 10, derived from the KiCad–OpenSCAD–FreeCAD–Blender Multi-software tasks. Tasks 02 and 09 have directly matched Multi-software counterparts and are used for the paired analysis below. Fig. C.2 summarizes their shared engineering requirements; the workflow and delivery differences are described afterward.

**Shared design, different delivery requirements.** For both tasks, the Multi-software instruction requires KiCad to extract the board geometry and parameters, OpenSCAD to construct the enclosure, FreeCAD to assemble the parts and check clearances, and Blender to produce a visual review. It specifies this sequence and 18 artifacts, including intermediate handoffs, reports, a rendered review, and a toolchain invocation log. The Open-ended instruction permits any available tools, scripts, or libraries in any order. It requires one final STEP assembly containing the enclosure, PCB, installed components, and task-specific features, which is evaluated directly against the geometric design requirements. The target assembly is shared; the required construction process and supporting deliverables change.

All seven models fail the Multi-software versions of tasks 02 and 09. On Open-ended task 09, however, Claude Opus 5, Gemini 3.7 Flash, Qwen 3.8 Max, and DeepSeek V4.1 Flash succeed; Open-ended task 02 remains unsuccessful for all seven models (Table C.3). The paired Gemini and Qwen 3.8 Max trajectories for task 09 locate the Multi-software failures and the changes in the successful Open-ended workflows.

**Where the Multi-software runs fail.** On task 09, Gemini submits a 478-byte `toolchain_invocation_log.json`, below the source evaluator's 500-byte minimum. Qwen 3.8 Max reaches its 150-step limit with the same required file missing. Its last recorded action checks the required artifacts after completing the Blender stage. Both runs fail the delivery requirements before their assemblies are checked for geometric validity.

**How the Open-ended runs succeed.** In the Gemini and Qwen 3.8 Max Open-ended task-09 trajectories, the agents construct the assembly through FreeCAD scripting and submit a STEP file that passes the final geometry evaluator. Qwen 3.8 Max completes this run in 30 steps under the same 150-step budget as its Multi-software run. Gemini completes its Open-ended run in 41 steps under a 150-step limit; its Multi-software run ends after 44 steps under a 150-step limit, with the log-size check still failing. Gemini therefore fails the Multi-software task on a delivery check despite remaining within its step budget.

For these two models, direct construction produces an assembly that passes the geometry checks, while the prescribed workflow fails on intermediate delivery obligations. Geometric errors remain a separate failure mode: GPT-5.6 Sol leaves 3.698 $mm^3$ of material in the Open-ended task-09 MH1 bore, above the 0.1 $mm^3$ tolerance. The paired outcomes separate two demands of the Multi-software task: constructing a valid assembly and completing the required software handoffs and documentation.

Table C.3: **Paired task outcomes.** Entries are binary, with 1 indicating success. Multi-software follows the prescribed workflow; Open-ended permits free tool and strategy selection.

| Model | Task 02 | | Task 09 | |
|---|---|---|---|---|
| | Multi-software | Open-ended | Multi-software | Open-ended |
| GPT-5.6 Sol | 0 | 0 | 0 | 0 |
| Claude Opus 5 | 0 | 0 | 0 | 1 |
| Gemini 3.7 Flash | 0 | 0 | 0 | 1 |
| Kimi K3 | 0 | 0 | 0 | 0 |
| Qwen 3.8 Max | 0 | 0 | 0 | 1 |
| Qwen 3.8 Flash | 0 | 0 | 0 | 0 |
| DeepSeek V4.1 Flash | 0 | 0 | 0 | 1 |

### C.4 Additional Ablation Analysis

Additional observations can change which tasks succeed as well as the quality of partially successful outputs. We separate these effects using matched task scores and examine the task requirements behind the opposing outcomes.

**Initial-image placement.** Providing the reference drawing in the message improves GUI EngiScore for all three models, while Kimi's CLI score decreases from 40.0 to 36.7 (Table 3). GPT's GUI gain combines 15 failures becoming successes and five successes becoming failures across 60 tasks. The gain is therefore a net result of different task outcomes, rather than an improvement shared by every task.

The opposing GPT outcomes on FreeCAD task 01 and Blender task 02 distinguish reference access from artifact construction. FreeCAD task 01, which becomes successful with message placement, requires a mounting plate whose drawing supplies dimensions and whose supplement supplies four through-hole locations. Direct presentation makes one of these two required information sources available at the start. Blender task 02 instead becomes unsuccessful. It requires ten courses of ten bricks as 100 disconnected islands within a single mesh object. Recognizing the wall's appearance leaves its mesh connectivity to be constructed correctly. These requirements explain why easier access to a reference can remove an information-retrieval step without ensuring that the resulting artifact satisfies its structural constraints.

**Runtime image access in CLI.** Enabling `readimg` improves Gemini on both task sets and lowers Kimi on both (Table 4). With a message-based initial drawing, Kimi gains two successful tasks but loses five. FreeCAD task 39 becomes successful, whereas task 42 becomes unsuccessful. Task 39 combines a bearing bore, a front window, top ribs, and mounting holes. Task 42 requires a central through-bore, four closed-end radial slots with specified depth, and a hole array with a specified angular phase. Both require exactly one valid connected solid.

A rendered view can support comparison of the visible arrangement of these features, but does not by itself establish bore connectivity, slot depth, or the validity of the exported solid. These properties require geometric checks alongside visual inspection. The task pair therefore illustrates a distinction between making visual feedback available and verifying the engineering constraints.

GPT's EngiScore increases from 66.7 to 68.3 with a message-based drawing but decreases from 48.1 to 41.5 without one. An initial drawing could give the agent a target for interpreting intermediate renders. However, the two settings contain different task sets: each supports a matched comparison of `readimg` on versus off, but their difference also includes task composition. The GPT reversal therefore motivates an interaction hypothesis rather than isolating the interaction between initial-image availability and runtime image access.

**Accessibility information.** Adding an accessibility tree improves Kimi's EngiScore but lowers Gemini's and GPT's (Fig. 6a). GPT has eight tasks gaining nonzero scores and eight losing them, yet its EngiScore decreases from 29.84 to 29.05. The gains contribute 7.195 points, while the losses remove 8.000 points. A further 0.014-point improvement on an already nonzero task only partly

offsets this deficit. Equal numbers of gains and losses therefore conceal unequal amounts of engineering credit.

Within FreeCAD, task 14 becomes successful with the tree, while task 20 becomes unsuccessful. Task 14 adds and fuses a cylindrical crossbar between two posts. Task 20 requires positioning an upright support, fusing it to a base, and cutting a through-hole along a specified axis. Textual interface information can help identify controls, but the requested operations still depend on the selected body, geometric placement, and feature direction. The opposing outcomes within one application show that the benefit cannot be reduced to whether that application exposes accessibility information; the relevant question is whether the information supports the particular geometric operation.

**Screenshot resolution.** Native resolution produces the highest EngiScore for all three models (Fig. 6b). For GPT, increasing resolution from $0.6\times$ to $1.0\times$ yields ten new full-score outcomes, contributing 10.000 points, while ten tasks lose a combined 9.140 points. This imbalance accounts for the increase from 28.98 to 29.84 despite the regressions. Kimi gains eleven nonzero outcomes and loses none, so its improvement is more consistent across tasks.

GPT's newly successful KiCad task 06 requires repairing a QFN-48 footprint's exposed pad, paste apertures, and courtyard while preserving the peripheral pads. Its small, closely spaced features make it a relevant case for improved visual discrimination. The newly unsuccessful Blender task 02 instead requires the 100 disconnected brick islands described above. Greater image detail may aid feature selection, but the mesh-structure requirement remains a separate construction obligation. The contrast distinguishes a task that requires selecting fine geometric features from one that requires constructing a particular mesh structure.

**History length.** Increasing history from ten to fifteen turns lowers Kimi's EngiScore from 26.21 to 20.12, while GPT changes little, from 36.61 to 36.74 (Fig. 6c). For Kimi, five tasks gain nonzero scores and eleven lose them, contributing $+4.368$ and $-10.532$ points, respectively. Changes between already nonzero scores add 0.073 points. The decline is thus dominated by tasks losing all credit rather than by small quality reductions across partially successful designs.

GPT's near-constant mean has a different explanation. Seven tasks gain nonzero scores, contributing 6.899 points, while seven formerly full-score tasks lose 7.000 points. ZBrush task 03, which remains nonzero in both settings, improves from 0.61434 to 0.843902 and contributes another 0.230 points. These three components account for the net 0.129-point increase. The task optimizes a gasket sealing-band surface under topology and coverage constraints, so the higher score records a quality improvement that a binary gain/loss count would miss. Its fifteen-turn score still falls below the initial design's 0.909292. Longer history therefore improves this output relative to the ten-turn run without establishing an improvement over the starting design, consistent with the distinction in App. C.2.

# D Case Studies

## D.1 Successful Workflow Cases

Across these four GUI runs, the requested edits are retained in the delivered files together with the required electrical, geometric, or image properties. All four runs pass their task-specific artifact checks and receive a score of 1.0 (Fig. D.1).

In KiCad, the `DATA0`–`DATA7` labels encode eight distinct nets that establish the required one-to-one MCU–SRAM pin mapping. The agent adds the two required power flags while retaining the existing components and power and control connections. Checks on the saved schematic verify both the new data connections and the preserved circuitry, with zero electrical-rule errors.

In AutoCAD, rectangular arrays express the repeated geometry through shared spacing parameters. After constructing the outline and slots, the agent combines $5 \times 3$ and $4 \times 2$ arrays with offset origins to produce 23 staggered tube holes. The exported DXF retains the required coordinates, radii, entity types, and layer assignments, preserving the drawing's geometric and organizational structure through export.

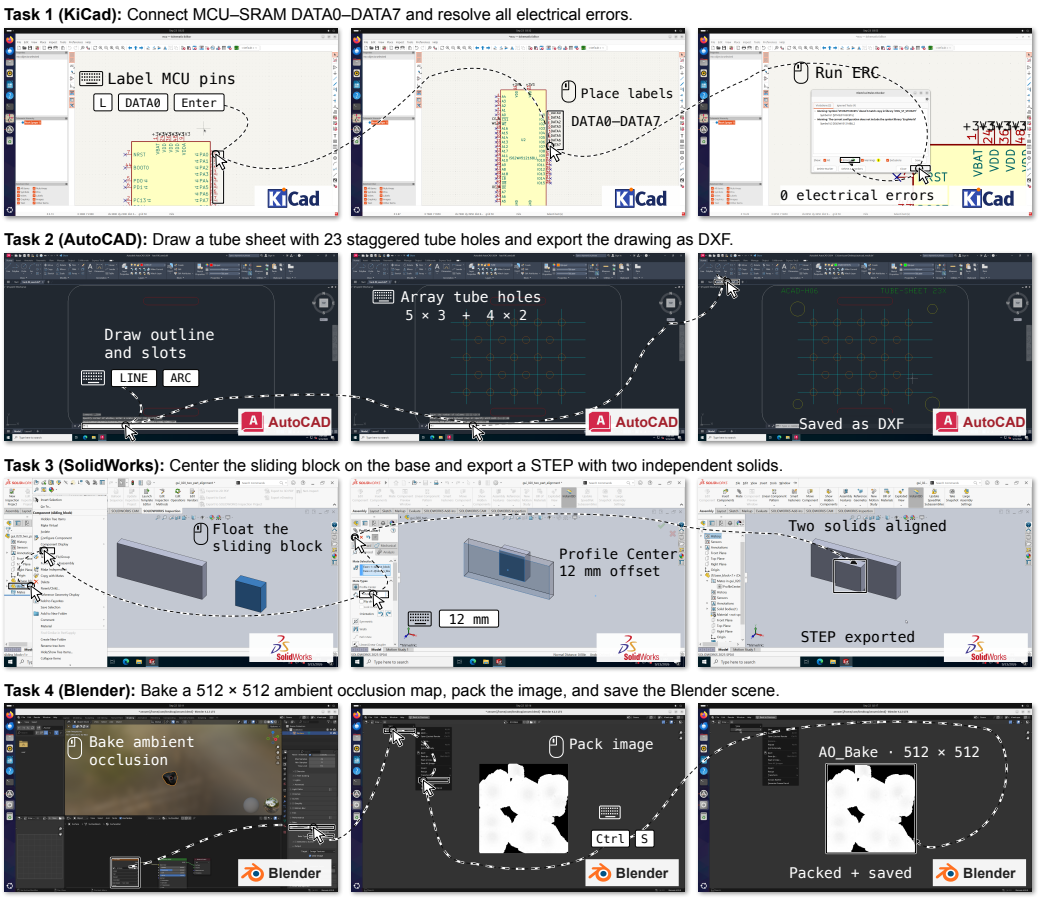



Figure D.1: **Successful single-software GUI workflows.** The four rows show representative KiCad, AutoCAD, SolidWorks, and Blender executions. Intermediate edits and final checks or artifacts are connected by dashed arrows. All four runs pass their artifact verifiers.

The SolidWorks edit changes relative placement while keeping part geometry fixed. The agent makes the sliding block movable, applies a Profile Center mate, and corrects the offset direction to align the XY centers and place the sliding block's bottom face against the base's top face. The exported STEP satisfies these placement constraints while retaining the original dimensions and two independent solids.

In Blender, the agent bakes ambient occlusion into `AO_Bake` and packs the $512 \times 512$ image before saving `answer.blend`. Packing embeds the baked pixel data in the native file, preserving the computed image as part of the delivered scene.

## D.2 Failure Cases

We analyze five recorded failure categories: DONE with a zero score, decision-turn exhaustion, explicit FAIL, environment timeout, and agent-initiated desktop closure or restart. Across the main evaluation and ablations, 3,347 distinct zero-score runs fall into these categories. Their aggregate distribution and main-evaluation breakdown are reported in Tables D.1 and D.2. Reused runs are counted once; required-field and missing-file failures are excluded from this analysis.

**Failure by interaction interface.** The failure composition differs substantially between interfaces (Fig. D.2). Among the analyzed failures, decision-turn exhaustion accounts for 233 of 636 CLI failures (36.6%) and 471 of 851 GUI failures (55.3%). Completion without acceptance is more common on CLI, while explicit failure declarations and runtime-limit failures remain smaller components in both settings. The interface-level breakdown complements the model-level analysis in Fig. 5.

Table D.1: **Failure taxonomy across all experiments.** Counts and shares of 3,347 distinct failed runs in the five analyzed categories. Reused runs count once; required-field and missing-file failures are excluded.

| Recorded outcome | Runs | Share (%) |
|---|---|---|
| DONE with zero score | 1,641 | 49.03 |
| Decision-turn limit reached | 1,499 | 44.79 |
| Explicit FAIL declaration | 167 | 4.99 |
| Environment time limit exceeded | 37 | 1.11 |
| Desktop closed or restarted by agent action | 3 | 0.09 |

Table D.2: **Failure counts by model.** Main-evaluation failures on the shared 300-task set, grouped into the five analyzed categories. Required-field and missing-file failures are excluded.

| Model | Analyzed | DONE | Turn limit | FAIL | Timeout | Desktop |
|---|---|---|---|---|---|---|
| GPT-5.6 Sol | 184 | 160 | 24 | 0 | 0 | 0 |
| Claude Opus 5 | 165 | 122 | 32 | 2 | 9 | 0 |
| Gemini 3.7 Flash | 222 | 88 | 133 | 0 | 0 | 1 |
| Kimi K3 | 226 | 122 | 94 | 10 | 0 | 0 |
| Qwen 3.8 Max | 221 | 84 | 131 | 4 | 0 | 2 |
| Qwen 3.8 Flash | 251 | 66 | 154 | 31 | 0 | 0 |
| DeepSeek V4.1 Flash | 218 | 53 | 136 | 1 | 28 | 0 |
| Total | 1,487 | 695 | 704 | 48 | 37 | 3 |

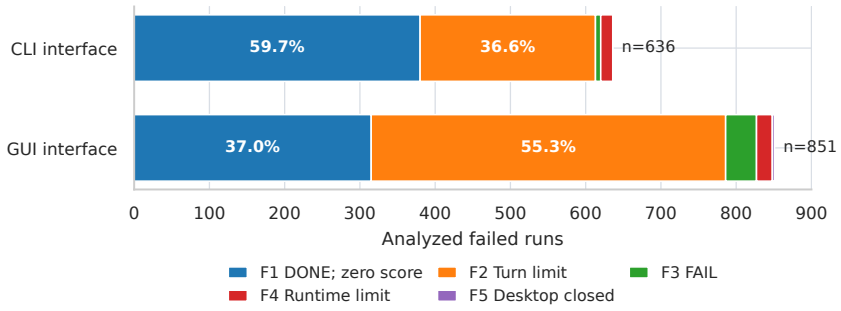



Figure D.2: **Failure composition by interaction interface.** Decision-turn exhaustion accounts for 36.6% of CLI failures and 55.3% of GUI failures; the remaining segments show the other four analyzed outcomes.

**Artifact checks in declared successes.** Independent checks identify geometric and cross-stage violations in four runs that end with `DONE` (Table D.3). The checks use the delivered artifacts even when agent-written reports claim that the requirements have been met.

Two of the four listed runs declare `DONE`; the remaining runs reach the 300-turn limit or declare `FAIL`. All four receive zero scores, with distinct discrepancies in the recorded operations and artifacts (Fig. D.3).

**Load representation mismatch.** In the SolidWorks–Abaqus case, the analysis completes and produces an ODB, but the native model uses a $-11.111$ MPa pressure load. On the specified $30 \times 3$ mm end face, this pressure corresponds to approximately 1 kN of tension. The evaluator checks native force or displacement parameters without converting pressure to resultant force, so this representation does not satisfy its load check.

**Import scale errors.** In the AutoCAD–SolidWorks–Abaqus case, an import-scale error becomes visible during extrusion. The agent corrects the units, but the error reappears after a crash and another DXF import. The transferred model is elongated and lacks the proportions of the $80 \times 60 \times 21$

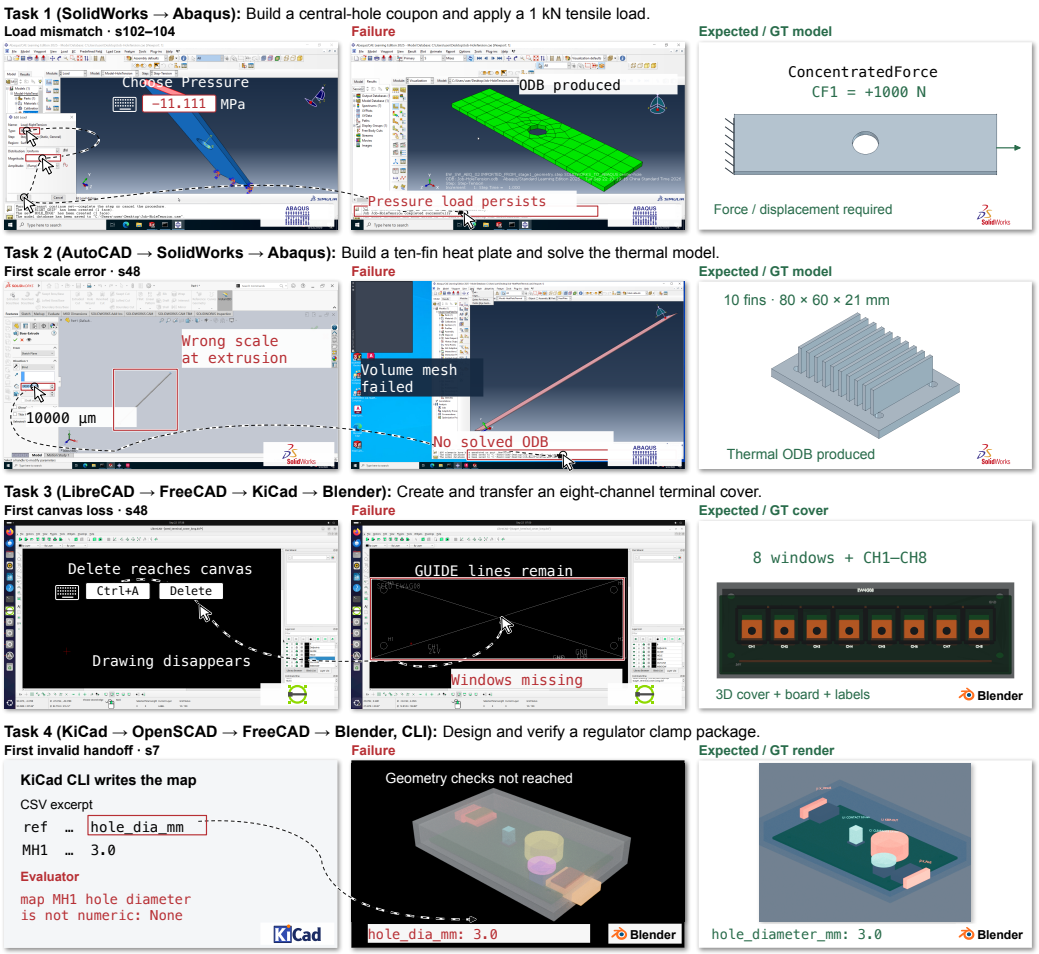



Figure D.3: **Multi-software failure cases.** Recorded actions, submitted data, and observed outcomes are compared with independently rendered reference artifacts. The first three cases use the GUI interface and the fourth uses the CLI interface; all receive zero credit.

Table D.3: **Verified violations after declared completion.** Each run ends with `DONE` but receives zero credit because the submitted artifact violates a task requirement.

| Task | Model | Verified violation |
|---|---|---|
| OpenSCAD microfluidic lid (GUI) | Gemini 3.7 Flash | Channel cut from the wrong face; sampled material and cavity states are inverted. |
| OpenSCAD V-groove pulley (CLI) | Gemini 3.7 Flash | Groove root and section fail at the required stations in the inherited datum frame. |
| KiCad–OpenSCAD–FreeCAD–Blender (CLI) | Gemini 3.7 Flash | The mechanical map assigns a mounting-hole role inconsistent with the source board. |
| Thermal-camera mount (Open-ended, CLI) | GPT-5.6 Sol | The MH1 bore contains 3.70 $mm^3$ of residual material, exceeding the 0.1 $mm^3$ allowance. |

mm, ten-fin reference. Repeated volume-meshing attempts continue until the 300-turn limit without producing a solved thermal ODB.

**Incorrect action targeting.** The LibreCAD–FreeCAD–KiCad–Blender case stalls during drawing preparation. An intended text edit sends `Ctrl+A` and `Delete` to the drawing canvas, clearing the drawing. After the seed drawing is reopened, the saved DXF still contains the diagonal GUIDE lines

and lacks the eight required windows. Subsequent attempts to launch FreeCAD do not advance the workflow, which ends with `FAIL`.

**CSV schema mismatch.** The KiCad–OpenSCAD–FreeCAD–Blender CLI case produces all 18 required files, including a final Blender render, and ends with `DONE`. Its mechanical CSV stores mounting-hole diameters under `hole_dia_mm`, whereas the GT uses `hole_diameter_mm`. The submitted field name is not among the evaluator's accepted aliases. The evaluator therefore reads the diameter of `MH1` as `None` and stops before the downstream geometry checks.