COURSEPROJECT-BENCH: EVALUATING AI AGENTS IN AUTHENTIC COURSE ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

AI agents are increasingly evaluated on long-horizon tasks involving tools, files, and complex environments, yet existing benchmarks typically rely on agents' pretraining knowledge or provide the necessary domain knowledge directly. Therefore, we introduce **CourseProject-Bench**, a benchmark of 30 university course projects spanning five disciplinary domains, each paired with a reconstructed hierarchical course workspace. These projects require **Active Knowledge Acquisition**: agents must access relevant instructional materials from a broader course environment and understand the conceptual or procedural knowledge they contain before applying it to the project. To evaluate these open-ended tasks, we introduce **Process Correctness**, a five-stage framework covering Task Understanding, Knowledge Access, Conceptual Understanding, Knowledge Operationalization, and Artifact Production. Across eight frontier agent configurations, the strongest achieves only 60.7% Process Correctness. Stage-level analysis reveals that Active Knowledge Acquisition remains challenging, with conceptual understanding emerging as a major bottleneck. We further find that model capability has a substantially larger effect than harness choice, while performance also varies across disciplines and artifact demands. CourseProject-Bench provides a diagnostic testbed for studying agents that must learn from complex environments and apply acquired knowledge to long-horizon tasks.

1 INTRODUCTION

Recent advances AI agents have enabled them to tackle increasingly long-horizon tasks involving tools, files, and complex environments. Existing benchmarks now evaluate whether agents can execute such workflows in realistic tasks (Jimenez et al., 2024; Xu et al., 2026b; Tang et al., 2026; Patwardhan et al., 2026). However, the tasks in these benchmarks typically either rely on agent's pretraining knowledge or provide the necessary domain knowledge in a direct and concentrated form. In many real-world settings, by contrast, the relevant knowledge is distributed a large and noisy workspace: agents must first locate the useful materials among many task-irrelevant files, then understand and apply the acquired knowledge to complete the task. This setting remains comparatively underexplored in existing agent evaluation.

University course projects provide a natural testbed for this setting. Like the tasks studied in existing agent benchmarks, they also feature long-horizon planning, tool use, interaction with complex workspaces, and heterogeneous artifact production. Furthermore, course projects can impose an additional requirement: the knowledge needed to complete the project may be distributed across a semester of lecture notes, readings, videos, assignments, code, and other instructional materials rather than provided directly with the project specification. Successful completion therefore requires agents not only to execute complex workflows, but also to locate the relevant materials, understand the underlying concepts, and operationalize them in concrete project artifacts, as depicted in Figure 2. We refer to the process of locating, accessing, and understanding the knowledge required by the project as **Active Knowledge Acquisition**. Evaluating agents on such *knowledge-dependent projects* therefore provides a unified test of both the long-horizon capabilities emphasized by existing agent benchmarks and the additional Active Knowledge Acquisition ability.

We introduce **CourseProject-Bench**, comprising 30 qualified university course projects spanning five disciplinary domains and a reliable evaluation framework. These projects preserve the broader

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.

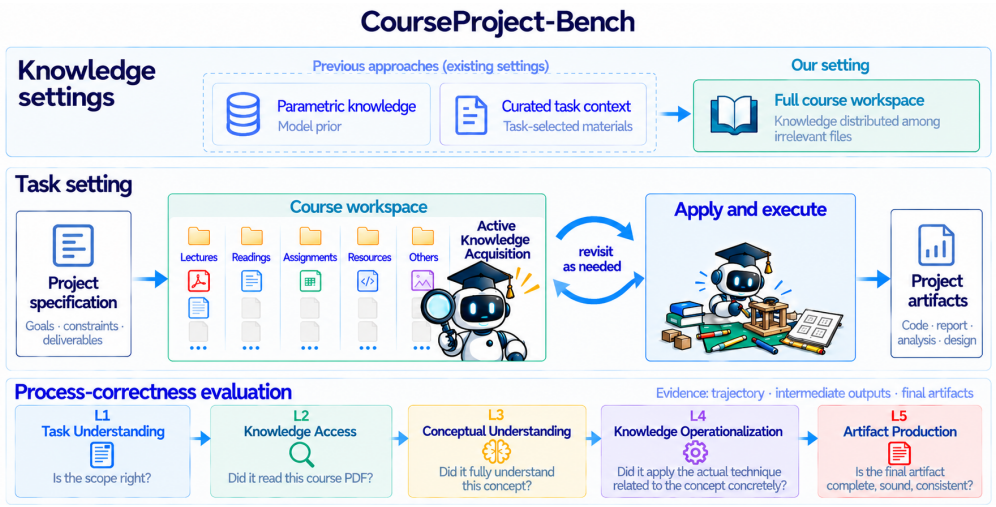


Figure 1: Overview of CourseProject-bench, with the unique knowledge setting and novel process-correctness evaluation.

challenges of long-horizon agentic work, while additionally requiring agents to perform **Active Knowledge Acquisition**. To ensure that, we apply a rigorous screening procedure that traces project requirements to the instructional knowledge needed to satisfy them and excludes projects that can be reliably completed from the specification or general background knowledge alone. For each retained project, we reconstruct the corresponding course workspace to preserve the original hierarchical file organization available to students, containing all lecture videos, pdfs, code, and supporting files.

Evaluating such project completions poses a separate challenge. To reduce the influence of subjective factors in evaluating open-ended course projects, we focus on whether an agent carries out the entire project completion process correctly rather than how good the final product appears. As depicted in Figure 2, we introduce a five-stage **Process Correctness** framework covering *Task Understanding*, *Knowledge Access*, *Conceptual Understanding*, *Knowledge Operationalization*, and *Artifact Production*. Notably, *Knowledge Access* and *Conceptual Understanding* correspond directly to the two core stages of **Active Knowledge Acquisition**. Verification over trajectories, intermediate evidence, and final artifacts, allowing us to separately characterize agent performance at each stage.

We evaluate eight frontier agent configuration, including Claude Opus 5, Claude Sonnet 5 and GPT-5.6-sol on their native harness, as well as DeepSeek-V4-Pro, GLM-5.3, Kimi-K2.7-Code, Qwen3.7-Max on Claude Code harness and Grok-4.6 on Opencode. The strongest agent achieves only **60.7%** Process Correctness on CourseProject-bench. Stage-level evaluation further localizes concrete failure modes, and shows that Active Knowledge Acquisition is a challenging setting, and the larger bottleneck lies in conceptual understanding. We also find that model capability affects Process Correctness substantially more than harness choice. Finally, performance varies considerably across disciplinary domains and declines as artifact demands increase, although neither factor alone fully explains agent performance.

Our contributions are threefold. First, to study the underexplored setting where the knowledge required for a task where the knowledge required for a task is distributed across a broad environment rather than provided as pre-selected task context, we introduce **CourseProject-Bench**, a benchmark of carefully screened university course projects paired with reconstructed instructional environments. Second, we introduce a five-stage **Process Correctness** evaluation framework that combines objective project-specific requirements with execution evidence to support scalable and fine-grained diagnosis of how agents perform during project completion. Third, we systematically evaluate eight frontier agent configurations and provide stage-level analyses that reveal their overall performance, key bottlenecks in Active Knowledge Acquisition, and the effects of harness choice, disciplinary domain, and artifact demands.

Table 1: Comparison of several agent benchmarks. A hierarchical workspace does not necessarily imply active knowledge acquisition if the knowledge acquired is merely task-related information such as file dependencies.

Benchmark	Task Source	Hierarchical Workspace	Active Knowledge Acquisition	Agent-Harness Combination	Process-correctness Evaluation
SWE-Bench (Jimenez et al., 2024)	GitHub issues	✓	✗	✗	✗
TheAgentCompany (Xu et al., 2026b)	O*NET inspiration	✓	✗	✗	✗
PaperBench (Starace et al., 2025)	Research papers	✗	✗	✗	✗
GDPval (Patwardhan et al., 2026)	Industry experts	✗	✗	✗	✗
AcademiClaw (Yu et al., 2026)	Students	✗	✗	✗	✗
Workspace-Bench (Tang et al., 2026)	Internet Company Roles	✓	✗	✓	✗
CourseProject-Bench	Authentic University Courses	✓	✓	✓	✓

2 Related Work

2.1 Agent benchmarks.

Existing agent benchmarks have progressively moved from isolated reasoning and tool-use tasks (Liu et al., 2024; Mialon et al., 2024) toward more realistic and extended interaction settings, such as web, enterprise, and desktop environments (Zhou et al., 2024; Drouin et al., 2024; Xie et al., 2024; Deng et al., 2023; Koh et al., 2024). More recent benchmarks extend evaluation to long-horizon tasks encountered in professional workplaces (Jimenez et al., 2024; Miserendino et al., 2025; Xu et al., 2026b; Merrill et al., 2026; Patwardhan et al., 2026) or scientific research (Huang et al., 2023; Chan et al., 2025; Siegel et al., 2024; Starace et al., 2025). AcademiClaw (Yu et al., 2026) further evaluates difficult student-originated academic tasks across diverse domains, while Workspace-Bench (Tang et al., 2026) focuses on reasoning over large hierarchical workspaces with explicit file dependencies.

None of these benchmarks, however, features **Active Knowledge Acquisition** during task completion. Existing settings typically miss one of its two components. Some rely on knowledge already encoded in model pretraining and require only task-specific information acquisition, such as locating files, dependencies, or workspace structure (Jimenez et al., 2024; Yu et al., 2026). Others provide the required knowledge directly through curated task-specific materials, removing the need for active source discovery (Yu et al., 2026). CourseProject-Bench requires both independent access of relevant knowledge sources and understanding of the conceptual or procedural knowledge they contain for later implementation.

2.2 Evaluation Design.

Existing agent benchmarks mainly evaluate task outcomes through tests (Jimenez et al., 2024), environment-state checks (Xu et al., 2026b; Tang et al., 2026), or rubric-based grading of submitted artifacts (Starace et al., 2025; Patwardhan et al., 2026). These designs provide effective measures of task completion, but offer limited visibility into agent behavior and intermediate process. Workspace-Bench provides the agent trajectory to the judge as supplementary evidence, but it does not assign rubric scores directly to the trajectory itself. Its rubric evaluation remains centered on the final workspace state, files, and artifact relationships. CourseProject-Bench instead evaluates **Process Correctness** across 5 workflow-aligned stages using trajectory evidence, intermediate artifacts, and final deliverables, enabling failures to be localized to specific stages, facilitating subsequent analysis and improvement.

3 The CourseProject Benchmark

3.1 Course Project Collection

Task Source. CourseProject-Bench is constructed from authentic university course projects from OpenCourseWare repositories and public university course websites. For sources such as MIT Open-

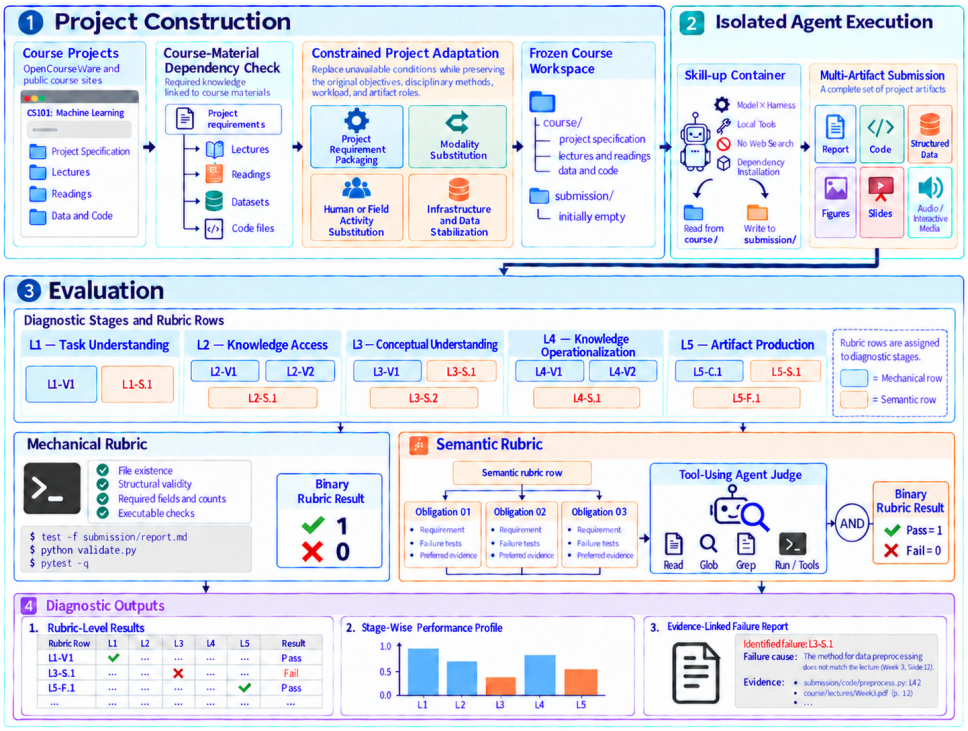


Figure 2: Overview of the construction and evaluation of CourseProject-bench.

CourseWare, existing project filters are utilized. For other institutions, systematic web search is launched through queries that combine university names with department names or disciplinary keywords. Each retrieved course page is individually inspected to retain only those that provide a substantial course project released under a compatible Creative Commons license.

Project Filtering for Active Knowledge Acquisition. CourseProject-Bench retains only projects with substantive dependence on course-specific knowledge. We first identify project specifications that explicitly reference taught concepts, such as a "three-stage framework from Lecture 10," and trace these references to the corresponding instructional materials. We identify a class of false positives that we term *concept pointers* for cases where the referenced material merely points to a standard concept already recoverable from general pretrained knowledge. For example, requirement such as "use a convolutional neural network from Lecture 8" normally does not create substantive course dependence unless the lecture introduces a novel or course-specific variant. Dependency annotations are produced by an annotation agent and are spot-checked by university student annotators. We additionally utilize an annotation model to generate a binary classification whether the project can be reliably completed given only project specification-related files. Judge results are inspected manually before removal. All together we explore 9k+ university courses, 455 courses with a valid project and license, and ultimately resulting in 30 qualified projects from 29 courses.

Hierarchical Workspace Preservation for Active Knowledge Acquisition. For each retained project, we reconstruct a local course fixture guided by two principles. *Content fidelity* preserves the complete and original course contents, including course webpages, lecture notes, readings, code, and other referenced materials. *Structural fidelity* preserves the original web-based hierarchy, organization, and navigation of the course, aligning the benchmark fixture with how students naturally browse, locate, and learn, rather than only keeping knowledge-providing files and flattening them into a benchmark-specific context. Each of the final 30 projects are paired with a course workspace containing a median of 87 files, 551 PDF pages, and 131K words, with an overall median hierarchy depth of 3. Additionally, 14 courses include a total of 282.4 hours of lecture video.

Task Adaption. To make heterogeneous course projects executable under a controlled agent environment, we apply a constrained adaptation procedure to requirements that depend on unavailable

conditions such as human interaction, physical-world access, specialized technical infrastructure or restricted external data. We therefore adapt projects only to remove such execution barriers while preserving the original task semantics and the competence being evaluated as closely as possible. Adaptations fall into four categories: *packaging*; *modality substitution*; *human/field activity substitution* and *infrastructure/data stabilization*. Examples are provided in Appendix A.2. All modifications are explicitly reviewed by university student annotators for consistency with the original course requirements.

3.2 Benchmark Characteristics

Broad disciplinary and task coverage. CourseProject-Bench consists of 30 authentic university course projects distributed relatively evenly across five broad disciplinary groups. This diversity enables us to evaluate agent performance across different forms of academic work, from computational and quantitative problem solving to source-grounded analysis, writing and design production.

Active knowledge acquisition. All 30 projects require prerequisite conceptual or procedural knowledge that is not fully specified in the project prompt and must be obtained from the Hierarchical course workspace. We further require acquisition to be *active*:the relevant knowledge sources are not provided as pre-selected task-specific context, but must be accessed from the broader course hierarchy. We evaluate *active knowledge acquisition* as two separable stages reflected in our evaluation framework: *knowledge access* (L2) and *conceptual understanding* (L3). This unique setting distinguishes CourseProject-Bench from tasks whose underlying methods are already familiar to agents from pretraining or knowledge-providing files are directly packed.

Long-horizon, multi-artifact projects. As authentic university course projects, the tasks are naturally designed for multi-week completion. A project contains a median of five deliverables, spanning reports, code, data, slides, figures, audio. 16 projects require executable code, and 23 include explicit cross-artifact production dependencies.

Process-Correctness Evaluation. Unlike conventional grading and existing agent benchmarks, which primarily assess final artifacts or task outcomes, CourseProject-Bench evaluates both the completion process and the resulting artifacts. We decompose project completion into five workflow-aligned stages and assess correctness at each stage, with final artifact quality captured as part of the process. This design exposes failure modes and helps localize where they arise, providing more diagnostic feedback than outcome-only evaluation.

3.3 Execution Environment

Each project runs in an isolated Docker container with the complete course workspace. Agents may install additional dependencies at runtime, but no per-project dependencies are preinstalled. Candidate agents operate cannot perform web search, as all task-relevant resources are materialized into the workspace beforehand. Execution and evaluation are strictly decoupled: evaluator-side artifacts, including rubrics, obligations, and deterministic checks, are introduced to the container only after candidate execution has completed.

4 Evaluation Framework

4.1 Process Correctness Evaluation

Traditional course grading schemes suffer from a major limitation when applied to agent evaluation. They often incorporate subjective criteria such as originality, aesthetics, creativity, or stylistic quality, which are unstable and unreliable. CourseProject-Bench therefore adopts a unified **Process Correctness** evaluation that assesses objective correctness throughout the entire project-completion process, from the agent's initial interpretation of the project specification to the submission of the final artifacts, while excluding subjective criteria.

We organize completion process into five workflow-aligned stages: **L1 Task Understanding** evaluates whether the agent correctly identifies the project scope, constraints, and required outputs. **L2 Knowledge Access** evaluates whether the execution trajectory demonstrates direct access to task-relevant instructional sources. **L3 Conceptual Understanding** evaluates whether the relevant dis-

ciplinary conceptual knowledge is understood correctly. **L4 Knowledge Operationalization** evaluates whether this knowledge is concretely instantiated in project decisions and artifacts. **L5 Artifact Production** evaluates the final submission along three objective and complementary dimensions: *completeness* (L5-C), *soundness* (L5-S), and *fidelity* (L5-F), covering whether required artifacts are present, substantively valid, and mutually consistent and traceable.

Together, L2 and L3 operationalize *active knowledge acquisition*. We intentionally separate these two stages because source access is observable evidence of acquisition effort but does not itself establish learning. The five stages characterize a logical dependency structure rather than a prescribed chronological trace. Agents may interleave these stages during execution, for example by returning to course materials after implementation reveals a knowledge gap.

4.2 Rubric Construction

Each of the five workflow stages is operationalized through project-specific rubric criteria derived from the project specification. Each rubric is evaluated against concrete execution evidence, including the trajectory, intermediate artifacts, and final deliverables. Although the evaluation is organized into five workflow stages, not every stage naturally produces a dedicated artifact that directly represents the corresponding capability. Some stage-level judgments must therefore be established from existing deliverables. The design that restricts evidence to naturally generated artifacts and avoid modifying project prompts for additional evaluation-specific outputs reduces rubric leakage. If agents are explicitly instructed to produce intermediate files for particular checks, they may polish or fabricate those files.

Rubric criteria are evaluated through either **mechanical** or **semantic** verification. Mechanical criteria are implemented as deterministic checks over observable properties such as file existence, structural validity, executable behavior, or numerical agreement. A semantic rubric captures a single project requirement that requires comprehensive evaluation. Direct rubric-level judgment is often error-prone: partial satisfaction or superficially plausible evidence may cause the judge to mark the criterion as satisfied despite an unmet sub-condition. We therefore decompose each semantic rubric into frozen verification obligations, each corresponding to one necessary sub-condition of the parent rubric. Each obligation specifies (i) the sub-condition that must hold, (ii) failure tests that guide targeted attempts to falsify that condition, and (iii) preferred evidence indicating where judging evidence is most likely to be found. Rubric examples can be found in Appendix A.3.

Rubric criteria and obligations are generated by an annotation agent and spot-checked by university student annotators. The audit checks that each criterion is traceable to authoritative sources and that neither rubrics nor obligations introduce, strengthen, or narrow the original requirements. Flagged items are revised and re-audited before the evaluation package is frozen.

4.3 Evaluation and Scoring

In practice, semantic criteria are evaluated with an Agent-as-Judge protocol as Tang et al. (2026). Each semantic rubric triggers a single judge invocation that receives all associated obligations, and the authorized evidence. The judge returns a binary verdict for every obligation. Each obligation receives 1 only when its required condition is verified; otherwise it receives 0. Let $\mathcal{P}$ denote the set of 30 projects and $\mathcal{C}$ the set of evaluated model–harness configurations. For project p, let $\mathcal{R}_p$ denote its set of verifiable rubric items, and let $r_{p,c,i} \in 0, 1$ indicate whether rubric item i is satisfied by configuration c on project p. The parent rubric passes only when all associated m_i obligations are satisfied:

$$r_{p,c,i} = \mathbb{I}\left[\bigwedge_{j=1}^{m_i} o_{p,c,i,j} = 1\right].$$

The Process Correctness Score for a single project-configuration pair ($\mathrm{PCS}_{p,c}$) is the fraction of satisfied rubric items, while the PCS for a single configuration (PCS_c) is the average of $\mathrm{PCS}_{p,c}$ across all projects:

$$\mathrm{PCS}_{p,c} = \frac{1}{|\mathcal{R}_p|} \sum_{i \in \mathcal{R}_p} r_{p,c,i}; \mathrm{PCS}_c = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathrm{PCS}_{p,c}.$$

Table 2: Main results of the 8 deployment-oriented configurarion on CourseProject-Bench. Process Correctness Score here is calculated as PCS_c from Section 4.3.

Model & Harness	Success Rate (%)	Process Correctness	Tool Calls	Time (min)
DeepSeek-V4-Pro + Claude Code	100.0	**60.7**	86.3	24.2
GPT-5.6-sol + Codex	100.0	59.9	**25.2**	**6.0**
GLM-5.3+ Claude Code	96.7	55.9	124.7	30.8
Kimi-K2.7-Code + Claude Code	96.7	48.4	84.2	36.9
Qwen3.7-Max + Claude Code	100.0	45.6	80.3	12.8
Claude-Opus-5 + Claude Code	100.0	40.0	47.8	16.6
Claude-Sonnet-5 + Claude Code	93.3	29.7	42.7	14.6
Grok-4.6 + OpenCode	96.7	22.1	28.7	8.9

Thus, each project contributes equally to the benchmark-level score regardless of the number of rubric it contains. The calculation of the other reported scores is described in the Appendix A.4.

5 Experiments

5.1 Experimental Setup

Models and Agent Harnesses. Our evaluation involves 12 agent configurations organized into two comparison groups. The first is a deployment-oriented comparison that uses native harness as much as possible: Claude Opus 5 and Sonnet 5 with Claude Code, and GPT-5.6-sol Singh et al. (2025) with Codex. DeepSeek-V4-Pro (Xu et al., 2026a), GLM-5.3 (Zeng et al., 2026), Kimi-K2.7-Code (Team et al., 2025), and Qwen3.7-Max Yang et al. (2025) are evaluated with Claude Code. Grok-4.6 was evaluated with OpenCode because it provides the compatible and stable performance.

The second is a controlled harness comparison that holds the underlying model fixed while varying the harness. We additionally evaluate Claude Opus 5 and GPT-5.6-sol with OpenCode and Hermes. This comparison estimates harness-related differences for these two models only, and it is not intended to provide a harness effect estimate for every model.

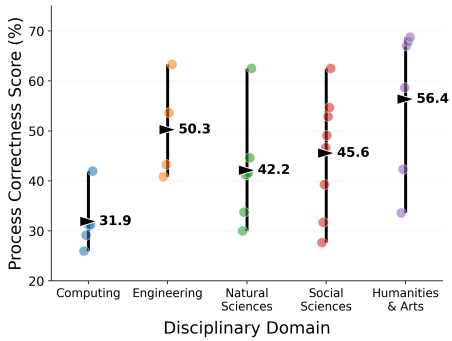
Execution infrastructure. Skill-Up serves as the common orchestration framework for all experiments. Every agent–project configuration runs inside an isolated Docker container. The image provides general-purpose system tools but does not pre-install project-specific dependencies. Agents may install any additional packages they need during execution. Each run has a common execution budget of 200 turns and two hours. Web-search and web-fetch tools are disabled.

Evaluation infrastructure. During execution, the tested agent can access only the course workspace, while all evaluation assets are withheld. After execution, Skill-Up freezes the submission and trajectory and introduces the evaluation assets into the container. For Agent-as-a-Judge, we use GPT-5.5 with Codex across all experiments.

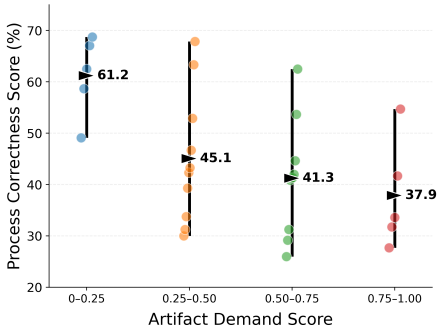
5.2 Main Results

CourseProject-Bench remains challenging among frontier agent configurations. Table 2 summarizes the main results for the eight deployment-oriented model–harness configurations. DeepSeek-V4-Pro with Claude Code achieves the highest process-correctness score (60.7), followed by GPT-5.6-sol with Codex (59.9) and GLM-5.3 with Claude Code (55.9). The remaining configurations score below 50, with Kimi-K2.7-Code at 48.4, Qwen3.7-Max at 45.6, Claude Opus 5 at 40.0, Claude Sonnet 5 at 29.7, and Grok-4.6 at 22.1. Success rate fails to achieve 100 for all configurations due to timeout, repetitive loops and unstable artifact generation. Importantly, Process Correctness is a correctness-oriented measure rather than a holistic project-quality score. It intentionally excludes subjective dimensions, and instead focuses on whether the project is completed correctly and faithfully according to requirements. Viewed against this relatively minimal bar, the best score of 60.7 still indicates substantial room for improvement.

More tool calls or longer time $\neq$ better performance. GPT-5.6-sol with Codex achieves runner-up performance with only 25.2 tool calls and 6.0 minutes per project. DeepSeek and GLM require



(a) Performance across disciplinary domains.



(b) Performance across Artifact Demand Score ranges, the four ADS group contains 5, 10, 10, 5 projects, respectively.

Figure 4: Process correctness across disciplinary domains and Artifact Demand Score ranges. Colored circles denote individual course-project scores, black markers denote group means, and vertical lines indicate the observed minimum-to-maximum range.

86.3 and 124.7 tool calls and 24.2 and 30.8 minutes, respectively, and each incur one timeout. Kimi-K2.7-Code incurs the longest runtime (36.9 minutes) without matching the top scores.

5.3 Harness Analysis

For the two models concerned, model capability affects process correctness substantially more than harness choice. From Figure3, GPT-5.6-sol exceeds Claude Opus 5 by 19.9–27.1 points across harness settings, while within-model harness variation is at most 3.8 points. In contrast, tool-call volume is highly harness-dependent: Opencode shows significantly more tool calls than native harness for Opus 5 and especially GPT-5.6, with about 4.8× as many tool calls as Codex. These results suggest that the model largely determines performance, whereas the harness primarily shapes interaction behavior, with more tool calls not necessarily translating into higher scores.

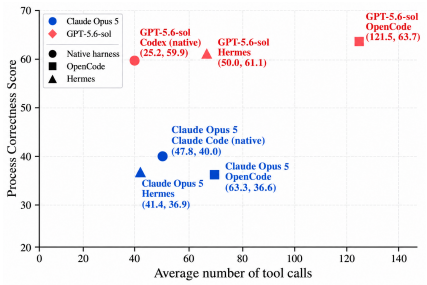


Figure 3: Harness Comparison on average score and number of tool calls.

5.4 Discipline-level Analysis

Agent performance varies substantially across disciplines, but these differences cannot be explained by disciplinary capability alone. At first glance, Figure 4a reveals substantial disciplinary variation in agent performance. Humanities & Arts achieves the highest mean process-correctness score (56.4%), while Computing obtains the lowest (31.9%). However, it is challenged by considerable within-discipline variation that high-mean domains still contain relatively low-scoring projects, and low-mean domains contain high-scoring outliers as well. Thus, gaps in disciplinary capability alone cannot fully explain agent performance.

To further explain this variation, we design Artifact Demand Score (ADS), a discipline-independent measure of project artifact complexity. ADS for each project is the average of four binary dimensions: whether it requires (1) executable artifacts, (2) integration across multiple components or artifacts, (3) explicit verification or validation, and (4) traceability between sources, computations, evidence, and conclusions. As shown in Figure 4b, mean process correctness declines from 61.2% in the lowest range to 37.9% as ADS increases. However, substantial overlap remains: for example, a mathematics project with modest ADS can still be difficult due to rigorous reasoning and derivation requirements. Together, these results suggest that **project difficulty is jointly shaped by**

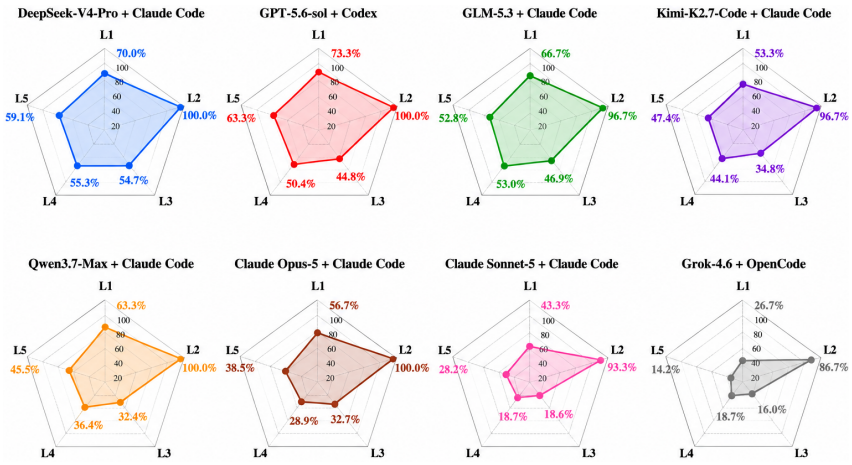


Figure 5: Stage-level performance across 8 agent configuration. Score for each stage ℓ is calculated as $\mathrm{PCS}_c^{(\ell)}$ from Appendix A.4.

disciplinary demands and artifact complexity, with neither factor alone fully explaining agent performance.

5.5 Stage-level Analysis

Active Knowledge Acquisition remains challenging at both stages, but the larger bottleneck lies in conceptual understanding. In Figure 5, agents generally perform well on Knowledge Access (L2) with four configurations achieving 100%, though source discovery is not fully solved. Illustrated by the trajectory, agents generally located and directly inspected the target materials or converted them into more readable formats. However, L3 scores fall sharply to 16.0–54.7%. The L3 failures suggest that agents often located the relevant instructional materials and concepts, but did not fully understand them. Their responses frequently captured only part of the required conceptual content, such as missing key mechanisms or assumptions, overlooking scope conditions or evidentiary distinctions.

Process-correctness evaluation exposes process–artifact discrepancies that may remain invisible in the final submission alone. This pattern is visible in Figure 5, where the L5 score of 6 configurations exceeds both L3 and L4. In other words, final artifacts can appear relatively successful even when the underlying conceptual understanding or knowledge operationalization is substantially weaker. For example, trajectory shows that a reported effect was actually fabricated through parameter selection. These checks do not prevent agents from making seemingly-pretty artifacts with errors, but they prevent such completions from being rated as successful project completion.

Five-stage process-correctness evaluation provides a structured diagnosis of failure compared to final-output-only evaluation. This enables failures to be traced to different stages clearly. Errors in different stages are provided in Appendix A.5. This diagnostic information can thus guide future agent improvement. Furthermore, several cases suggest that five stages behave cumulatively: earlier errors can carry forward and compound later failures, making early-stage correctness particularly important.

6 Conclusion

We introduce CourseProject-Bench, a benchmark of 30 authentic university course projects across five disciplinary domains for evaluating agents in Active Knowledge Acquisition settings. By preserving complete hierarchical course environments, the benchmark requires agents locate, understand and operationalize the domain knowledge. We further propose a five-stage Process Correctness evaluation framework that enables fine-grained diagnosis beyond final-artifact quality. Experiments across frontier model–harness configurations show that current agents remain far from reliably completing these projects, with a particularly large gap between knowledge access and understanding.

AI USE STATEMENT

In this work, we used generative AI tools for language editing and improving the presentation of the manuscript. All AI-assisted content was reviewed by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All 30 courses included in CourseProject-Bench were carefully reviewed to ensure that their materials are distributed under Creative Commons or other compatible open licenses.

REFERENCES

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025.

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. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023.

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? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

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.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Samuel Miserendino, Michele Wang, Tejal Patwardhan, and Johannes Heidecke. Swe-lancer: Can frontier llms earn $1 million from real-world freelance software engineering? *arXiv preprint arXiv:2502.12115*, 2025.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simon Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, et al. Gdpval: Evaluating ai model performance on real-world economically valuable tasks. In *International Conference on Learning Representations*, volume 2026, pp. 24005–24040, 2026.

Zachary S Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. Corebench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *arXiv preprint arXiv:2409.11363*, 2024.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. Paperbench: evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.

Zirui Tang, Xuanhe Zhou, Yumou Liu, Linchun Li, Yukai Wu, Weizheng Wang, Hongzhang Huang, Wei Zhou, Jun Zhou, Jiachen Song, et al. Workspace-bench 1.0: Benchmarking ai agents on workspace tasks with large-scale file dependencies. *arXiv preprint arXiv:2605.03596*, 2026.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026a.

Frank Fangzheng Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. Theagentcompany: benchmarking llm agents on consequential real world tasks. *Advances in Neural Information Processing Systems*, 38, 2026b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Junjie Yu, Pengrui Lu, Weiye Si, Hongliang Lu, Jiabao Wu, Kaiwen Tao, Kun Wang, Lingyu Yang, Qiran Zhang, Xiuting Guo, et al. Academiclaw: When students set challenges for ai agents. *arXiv preprint arXiv:2605.02661*, 2026.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

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

Table 3: Examples of project adaption.

Category	Course Project	Adaptation
Packaging	**Operating Systems**	The original project was distributed across five checkpoint documents. We consolidated CP1–CP5 into a single benchmark specification, while retaining the original APIs, hardware assumptions, point-valued requirements, timing bound, and event-log requirements.
Modality substitution	**Operating Systems**	The original assignment required screenshots from the EdSim51 GUI. Because the GUI was not available in the controlled environment, screenshots were replaced with annotated s51 execution transcripts that record the same observable events, such as thread creation, timer interrupts, context switches, semaphore states, and parking events.
Human/field activity substitution	**Documentary Studies**	The original project assumed interviews, photographs, recordings, or other fieldwork. We replaced these activities with a fixed archival packet containing life-history manuscripts, photographs, census records, catalog metadata, and rights information.
Infrastructure/data stabilization	**Media Education and the Marketplace**	To make the case-study evidence base reproducible, we operationalized the assignment with five fixed case dossiers drawn from public program documents, independent evaluations, academic studies, and documented testimony. The dossiers provide a stable source base while preserving the original requirements for case selection, evidence evaluation, impact analysis, and traceable citation.

A Appendix

A.1 Limitation and Future Work

First, CourseProject-Bench currently contains 30 authentic university course projects spanning five disciplinary domains. While these projects already cover diverse task structures, artifact types, and course environments, the benchmark remains limited in scale and is restricted to courses whose materials and projects are publicly accessible and legally redistributable. Expanding to additional institutions, disciplines, and project formats would further improve the benchmark's coverage and representativeness. Second, all reported results are based on a single-attempt evaluation protocol. Future work could introduce multi-trial evaluation to better characterize run-to-run robustness and agents' ability to recover from intermediate failures. Lastly, although we evaluate 12 model–harness configurations, controlled harness comparisons are currently available for only two underlying models. Extending the benchmark to a broader set of frontier models and systematically crossing models with different agent harnesses would enable more reliable analysis of model–harness interactions and help maintain a timely and representative evaluation suite.

A.2 Project Adaption Examples

Examples can be found in Table 3. Each adaptation was reviewed for necessity and scope. It removes only an execution barrier and retains the project's essential task semantics, competence requirements, and key assessment elements.

A.3 Rubric Examples

Rubric examples can be found in Table 4. The obligation subset is chosen due to limited space to represent part of the complete obligations to fulfill that rubric.

A.4 Scoring Equation for Reported Results

For stage-level analysis, let $\mathcal{R}_{p,\ell} \subseteq \mathcal{R}_p$ denote the rubric items assigned to stage $\ell \in L1, \ldots, L5$. The PCS for that stage of a configuration on one project ($\mathrm{PCS}^{(\ell)}_{p,c}$), and all projects are the following:

$$\mathrm{PCS}^{(\ell)}_{p,c} = \frac{1}{|\mathcal{R}_{p,\ell}|} \sum_{i \in \mathcal{R}_{p,\ell}} r_{p,c,i}; \mathrm{PCS}^{(\ell)}_{c} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathrm{PCS}^{(\ell)}_{p,c}.$$

Finally, when analyzing properties of individual projects, such as disciplinary variation or artifact demands, we aggregate across model–harness configurations:

Table 4: Rubric examples for 5 stages from different course projects. The obligation subset is chosen due to limited space to represent part of the complete obligations to fulfill that rubric.

Stage	Course Project	Rubric Type	rubric criterion	**Evidence**	**Representative obligation subset**
L1	Software Studio	S	"The trajectory or submitted design document demonstrates that, before substantial implementation, the agent identified the shopping-cart project from course/pages/projects/index.html, the chosen extension option, and the load-bearing constraints (local execution, shopper vs shopkeeper roles, persistent orders with price snapshots)."	trajectory sub:design.md	O1: Correctly identify the shopping-cart project from course/pages/projects/index.html. O2: Select exactly one legal extension option. O3: State that orders persist and that prices are snapshotted at order time rather than recomputed from current catalog prices.
L2	Projection Theory	S	"The trajectory contains genuine source-access events (direct reads or faithful, traceable derived representations such as extracted text, rendered pages, or readable sidecars) for the formal specification and the source groups required for the submitted work."	trajectory	O1: The required course sources for the selected direction were accessed through direct reads or faithful derived representations.
L3	Labor Economics	S	"The report correctly explains, in study-specific terms, observed selection bias in OLS returns to schooling, why college proximity (nearc4) is proposed as an instrument, the role of the first stage, and how 2SLS combines them; and it states the relevance, exclusion, and exogeneity assumptions required for a causal interpretation."	sub:report.md course:card-w4832.pdf	O1: Explain selection bias in the OLS schooling-return estimate in Card's setting. O2: Explain why nearc4 is proposed as an instrument. O3: Explain how the first stage feeds into the 2SLS estimate. O4: State and correctly map relevance, exclusion, and exogeneity to the study.
L4	Projection Theory	S	"The manufactured-solution study computes discretization errors in a stated norm on multiple resolutions, fits an observed order from generated values, and claims the expected order only when the implemented operator, boundary data, exact forcing, and refinement regime support that comparison."	sub:code/**/*.py sub:code/results_*.json sub:code/*.log sub:REPORT.md	O1: The exact solution, forcing term, boundary data, and implemented operator are mathematically consistent. O2: Errors are computed on at least two distinct resolutions under a defined refinement regime. O3: The observed convergence order is fitted from generated error values rather than merely asserted. O4: A producing path links code execution to the recorded errors and fitted order. O5: The operator, forcing, boundary data, and exact solution agree across the code and report.
L5	Labor Economics	M	"The canonical report, analysis script, README, table directory, and figure directory are present and non-empty."	sub:report.md sub:analysis.py sub:README.md sub:output/tables/**/* sub:output/figures/**/*	N/A

Table 5: Core errors concluded from experiment results by the stage-level evaluation.

Eval Stage	Core error	Typical Behavior	Examples
L1	**Requirement Misinterpretation**	Requirement omissions; Task narrowing; Constraint neglect	Course: Projection Theory The task asked for a result about the full set X, but the submission proved a weaker result only for a dense subset $X' \subset X$.
L3	**Conceptual Misunderstanding**	Conceptual confusion; Assumption violations; Method misuse	Course: Rock Climbing The model confused the required concept of friction along the rope path with the material's own damping property.
L4	**Incomplete Knowledge Operationalization**	Implementation gaps; Method substitution; Parameter neglect	Course: Error-Correcting Codes The project required a specified two-stage turbo decoder, but the submission used a more generic and simpler belief-propagation approach.
L4	**Insufficient/Circular Verification**	Circular validation; Missing sanity checks; Weak reproducibility	Course: Projection Theory The search procedure recomputed a closed-form expression already derived in the paper, so it did not independently test the claimed result.
L5	**Fabrication**	Evidence fabrication; Result invention; Citation fabrication	Course: Project Lab in Mathematics The paper claimed that the computation had been rerun up to 2×10^5, but the submitted script stopped at $p = 97$.
L5	**Incomplete or Inconsistent Final Artifacts**	Missing deliverables; Artifact incompleteness; Cross-artifact contradiction	Course: Project Lab in Mathematics Several values in Table 4 of the paper contradicted the submitted base_comparison.csv.

$$\overline{\mathrm{PCS}}_p = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathrm{PCS}_{p,c}.$$

This produces one configuration-averaged score per project and prevents projects from being implicitly replicated when analyzing project-level characteristics.

A.5 Error Types and Examples

Table 5 lists core errors concluded from experiment results by the stage-level evaluation.

A.6 Takeaway for Project Designers

Under our single-attempt, no-interaction protocol, current agents were unable to reliably complete complex course projects without process-level failures. This suggests a design principle for assignments intended to discourage uncritical agent substitution: increase artifact demands by requiring executable, integrated, independently verifiable, and traceable deliverables. In that case, weaknesses in planning, implementation, verification, and cross-artifact coordination become more difficult to conceal.

Evaluation should require process-bearing evidence in addition to the final deliverables when the process is relevant to correctness. Logs, intermediate data, execution records, versioned outputs, and verification reports can help identify unsupported completion claims, fabricated experiments, and inconsistencies that may not be visible in a polished final submission.