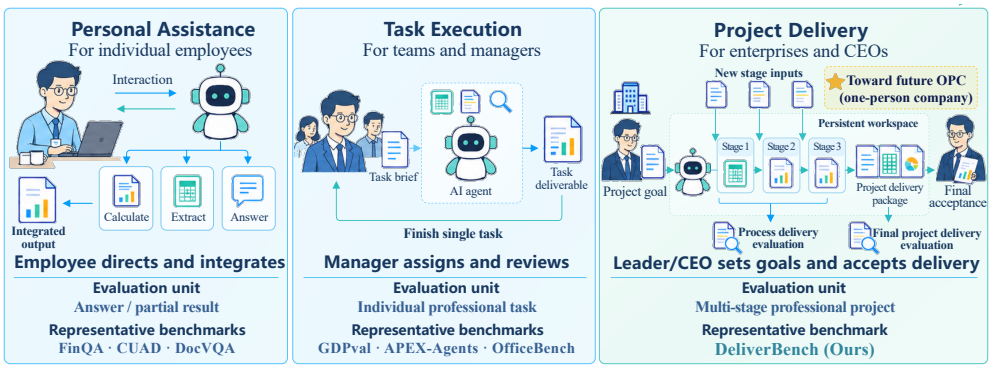
DELIVERBENCH: BENCHMARKING AI AGENTS ON DELIVERING REALISTIC MULTI-STAGE PROFESSIONAL PROJECTS

Anonymous authors
Paper under double-blind review

ABSTRACT

As LLM agents grow more capable, delegation is expanding from individual assistance to end-to-end professional project delivery, with potential to support enterprise automation and even one-person companies. Some existing benchmarks cover professional tasks and extended workflows, but inadequately assess how projects evolve and whether their final multi-file deliverables meet acceptance criteria. Thus, we introduce **DeliverBench**, which takes a multi-stage professional project as its evaluation unit. Reconstructed from real-world projects with domain experts, it includes 11 projects across seven domains and 150 interdependent subtasks. Agents work in a persistent workspace, incorporating new materials and client feedback released along a project timeline while reusing, revising, and integrating their own earlier artifacts. Expert-written rubrics assess checkpoint submissions and evaluate final packages for deliverable quality, cross-file consistency, and package acceptance. Across seven model–harness systems, we find that mean final-delivery scores range from 48.1 to 70.0 out of 100, and subtask pass rates at a 90-point threshold range from 13.3% to 50.0%. Every system's mean final-delivery score falls below its mean process score, revealing a gap between local task performance and project delivery. Correctness and usability are the weakest dimensions, while case studies highlight sustained quality and artifact repair. Runtime and token use vary widely, with greater expenditure not consistently associated with better delivery. These findings indicate that task-level scores alone cannot establish project readiness, motivating evaluation of agents' ability to sustain professional work and deliver coherent, usable packages.



How well can agents deliver multi-stage professional projects end to end?

Figure 1: **From personal assistance to project delivery.** DeliverBench extends evaluation from individual outputs and tasks to multi-stage professional projects, assessing both intermediate deliverables and the final multi-file delivery package.

1 INTRODUCTION

As AI agents become capable of producing professional work products (Patwardhan et al., 2026; Xu et al., 2026), the scope of delegation is expanding from assistance for individual employees toward

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.

Table 1: Comparison of DeliverBench with existing agent benchmarks. ✓ indicates full support for a feature, △ indicates partial support, and × means the feature is not specified in the benchmark's published protocol. Appendix A explains the comparison criteria.

Benchmark	Evaluation unit	Task characteristics				Evaluation design	
		Evolving inputs/feedback	Long-horizon tasks	Explicit subtasks	Cross-task artifact dependencies	Intermediate evaluation	Project-level delivery
GDPval (2026)	Occupational task	×	✓	×	×	×	△
APEX-Agents (2026)	Professional task	×	✓	×	×	×	△
OfficeBench (2024)	Office task	×	✓	△	△	×	△
WorkArena++ (2024)	Composite workflow	×	✓	✓	△	✓	×
TheAgentCompany (2026)	Workplace task	✓	✓	△	△	✓	△
PaperBench (2025)	Replication project	×	✓	△	△	△	✓
StartupBench (2026)	End-to-end workflow	×	✓	×	×	×	△
Workspace-Bench 1.0 (2026)	Workspace task	×	✓	×	×	△	△
SQBench (2026)	Delivery task	×	△	×	×	△	△
DeliverBench	**Multi-stage professional project***	✓	✓	✓	✓	✓	✓

*DeliverBench instances are reconstructed from real-world professional projects.

end-to-end delivery for enterprises. Figure 1 illustrates three modes of participation: a personal assistant supplies partial results that an employee integrates; a task executor completes assignments that a manager reviews; and a project-delivery agent carries an entire project from its initial brief to final handoff. In the last setting, clients define project goals and acceptance requirements, while agents take responsibility for the intervening work. Evaluating this form of delegation requires examining how agents carry outputs through interdependent tasks toward a shared objective.

Enterprise delivery requires more than completing individual assignments. In business-to-business (B2B) services, clients often commission a project whose deliverables must jointly serve a business purpose. An analysis becomes input to a later report; new evidence or client feedback may require revising both the analysis and a presentation based on it. Professional work therefore unfolds through evolving information, intermediate handoffs, and revisions across files. Managing these dependencies is a core coordination problem (Malone & Crowston, 1994). Even individually polished outputs can fail as a delivery package if they omit required components, disagree on key facts, or retain superseded conclusions. The client ultimately needs a complete, consistent, and usable package that satisfies the project's acceptance requirements.

Existing benchmarks capture parts of this setting, but leave project delivery underexplored (Table 1). Professional-task benchmarks such as GDPval, APEX-Agents, and OfficeBench assess whether agents can complete bounded assignments (Patwardhan et al., 2026; Vidgen et al., 2026; Wang et al., 2024). Other benchmarks incorporate workplace environments, file dependencies, research projects, or end-to-end workflows (Xu et al., 2026; Tang et al., 2026; Starace et al., 2025; Zhu et al., 2026). Interactive benchmarks also study policy-guided tool use (Yao et al., 2024) and proactive assistance across sessions (Zhang et al., 2026). Two requirements must be addressed together: *project simulation*, in which later tasks inherit and revise the agent's own artifacts as new information arrives; and *delivery evaluation*, which assesses intermediate work as well as the joint requirements of the final package. Our central question is: *How well can AI agents deliver multi-stage professional projects end to end?*

We introduce **DeliverBench**, a benchmark that takes a *multi-stage professional project* as its evaluation unit. Reconstructed from real-world projects with domain experts, it comprises 11 projects across seven professional domains, with 150 interdependent subtasks and 111 required final deliverables (Section 3.4). Agents work in a persistent workspace along a project timeline: task instructions, new materials, and client feedback arrive at prescribed stages, and later tasks consume, revise, or integrate earlier outputs. Each project culminates in a multi-file package governed by shared delivery requirements. The challenge is therefore to maintain and update accumulated work across tasks, beyond completing any one lengthy assignment.

DeliverBench pairs *process delivery evaluation* with *final project delivery evaluation*. Expert-written rubrics assess checkpoint submissions against each subtask's requirements and the information available at that point. The final package is evaluated through three complementary targets: *deliverable quality* of individual files, *cross-file consistency* among them, and *package acceptance*

against overall handoff requirements. Rubric criteria also carry quality-dimension labels, supporting diagnosis of correctness, completeness, consistency, professionalism, and usability. Process and final scores remain separate, while preserved submissions and execution records support tracing how earlier work is reused, repaired, or left unresolved.

Our contributions are follows.

- **A benchmark of real-world professional projects.** We reconstruct projects supplied by domain experts as staged, interdependent workflows in a persistent workspace, testing whether agents can carry their own artifacts forward toward a shared multi-file delivery objective.
- **Evaluation beyond individual deliverables.** We connect checkpoint assessment with final-package evaluation and diagnostics across five quality dimensions, capturing cross-file relationships and package acceptance requirements that isolated artifact scores leave untested.
- **Evidence of limited end-to-end delivery capability.** Across seven model–harness systems, mean final-delivery scores range from 48.1 to 70.0, and subtask pass rates at the 90-point threshold range from 13.3% to 50.0%. Every system exhibits a delivery gap: its mean final-delivery score is lower than its mean process score. Dimension profiles identify correctness and usability weaknesses, while case studies highlight the need to sustain quality and repair earlier artifacts.

2 Related Work

Table 1 compares representative designs; we discuss three related lines of work below.

Professional and office tasks. GDPval (Patwardhan et al., 2026) evaluates occupational work products, APEX-Agents (Vidgen et al., 2026) studies professional-service assignments in tool-rich worlds, and OfficeBench (Wang et al., 2024) covers cross-application workflows, including file creation and subsequent use. Agents' Last Exam (ALE) (Sun et al., 2026) evaluates expert-authored, long-horizon professional tasks involving real software and verifiable outputs. DeliverBench extends the evaluation unit to a multi-stage professional project, where outputs from one assignment become inputs to later assignments. This tests whether task-level competence produces artifacts that remain usable in subsequent work toward a shared business objective.

Workflow and project benchmarks. WorkArena++ (Boisvert et al., 2024) composes executable enterprise subtasks with state dependencies, while TheAgentCompany (Xu et al., 2026) combines workplace tasks with intermediate milestones and feedback from simulated colleagues. PaperBench (Starace et al., 2025) evaluates complete research-replication repositories by executing them in a fresh environment, providing project-level functional acceptance. StartupBench (Zhu et al., 2026) assesses final deliverables from long-horizon professional workflows; Workspace-Bench (Tang et al., 2026) models input-file dependencies and restores its workspace between independent tasks. In interactive settings, τ-bench (Yao et al., 2024) tests agents' API use and policy adherence through conversations with simulated users, while π-Bench (Zhang et al., 2026) evaluates proactive intent resolution and task completeness across sessions sharing a persistent workspace. DeliverBench makes project evolution explicit: instructions, new evidence, and client feedback arrive along a prescribed timeline, while later tasks inherit the agent's own artifacts. This tests coordinated revision: new information must be incorporated into earlier work and the dependent deliverables, keeping the project consistent as it evolves.

Process and delivery evaluation. AgentBoard (Ma et al., 2024) measures incremental progress using state matching and subgoal completion. TheAgentCompany provides checkpoint scores, and DevAI's Agent-as-a-Judge evaluates development milestones linked by a requirement DAG (Xu et al., 2026; Zhuge et al., 2024). PaperBench measures partial replication through a final-submission rubric; Workspace-Bench and SQBench also use trajectory evidence for process or risk checks (Starace et al., 2025; Tang et al., 2026; Sun, 2026). These mechanisms need not preserve and separately score submissions along the original project timeline. DeliverBench preserves checkpoint submissions and evaluates them separately from the final package, whose assessment covers deliverable quality, cross-file consistency, and package acceptance. This supports tracing whether earlier defects persist or are repaired, while cross-file and package checks reveal inconsistencies or omissions that individual-file scores cannot establish.

3 DELIVERBENCH

DeliverBench evaluates professional project delivery through intermediate submissions and a complete, consistent, and usable final package.

3.1 PROFESSIONAL PROJECTS AS THE EVALUATION UNIT

A *professional project delivery instance* specifies a professional role, a shared business objective, initial materials, interrelated subtasks, and a final multi-file package. Each project combines three elements:

- **A shared delivery contract.** Scope and acceptance requirements define what must be handed over. Individual files must meet professional standards and jointly satisfy package completeness, cross-file consistency, and the project objective.
- **Interdependent work in a persistent workspace.** Subtasks have distinct instructions and outputs that later work consumes, integrates, or revises. Checkpoints preserve submissions; stages organize delivery milestones and may contain several subtasks.
- **An evolving project timeline.** New materials, client decisions, and scripted feedback arrive according to the project schedule, requiring updates to earlier work and affected deliverables.

Appendix B.1 illustrates these elements through the task sequence and artifact dependencies of P05.

3.2 PROJECT SOURCING AND BENCHMARK CONSTRUCTION

Figure 2 demonstrates the five-step process for reconstructing real-world projects as expert-reviewed benchmark instances.

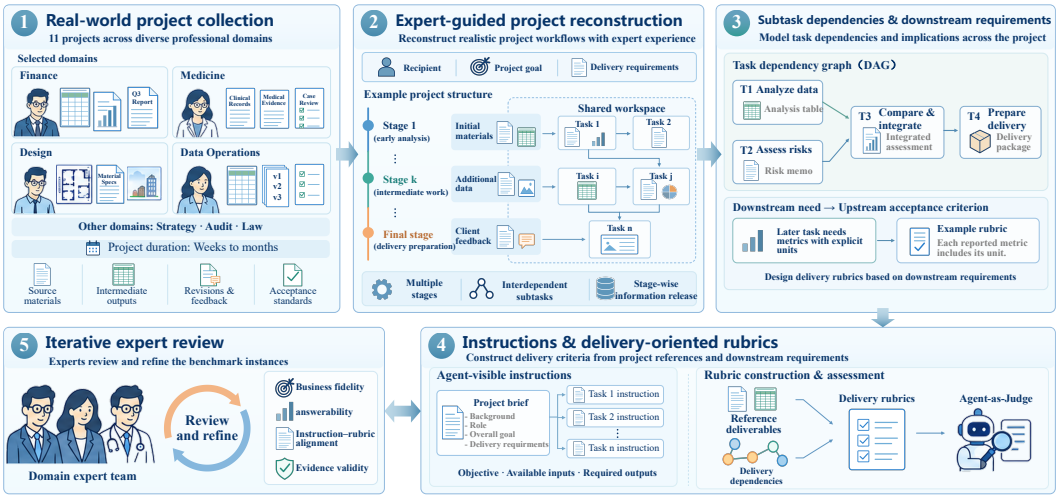


Figure 2: **Expert-guided construction of DeliverBench.** Real-world materials are reconstructed into staged workflows. Artifact dependencies and downstream requirements guide instruction and rubric writing, followed by expert review of business fidelity, answerability, and evidence validity.

1. Collect real-world project materials. Domain experts (e.g., lawyers and investment bankers with several years of professional experience) provide projects spanning weeks to months, together with source records, intermediate and final deliverables, feedback, and acceptance standards.

2. Reconstruct the project workflow. Experts help translate project scope and milestones into stages and subtasks with specified inputs, outputs, and information releases. Original records are distinguished from author reconstructions and scenario assumptions.

3. Model dependencies and downstream requirements. We identify how later tasks use earlier artifacts and which outputs a revision affects. These downstream needs determine what upstream deliverables must contain and inform their acceptance criteria.

4. Write instructions and delivery rubrics. The project brief and task instructions specify objectives, available inputs, and required outputs. Reference artifacts and dependency requirements inform task, cross-file, and package rubrics; evaluator-only references and future-stage materials are kept separate from currently visible inputs.

5. Review and refine with domain experts. Experts check business fidelity, factual support, stage-specific information sufficiency, and instruction–rubric alignment. Unresolved evidence gaps require repair or exclusion.

3.3 Execution and Evaluation Protocol

DeliverBench evaluates interdependent tasks as one continuous project. Figure 3 contrasts this setting (right) with a typical task-level setup (left), where each task is executed and scored independently. In DeliverBench, the same agent carries its artifacts forward across tasks toward a shared project objective.

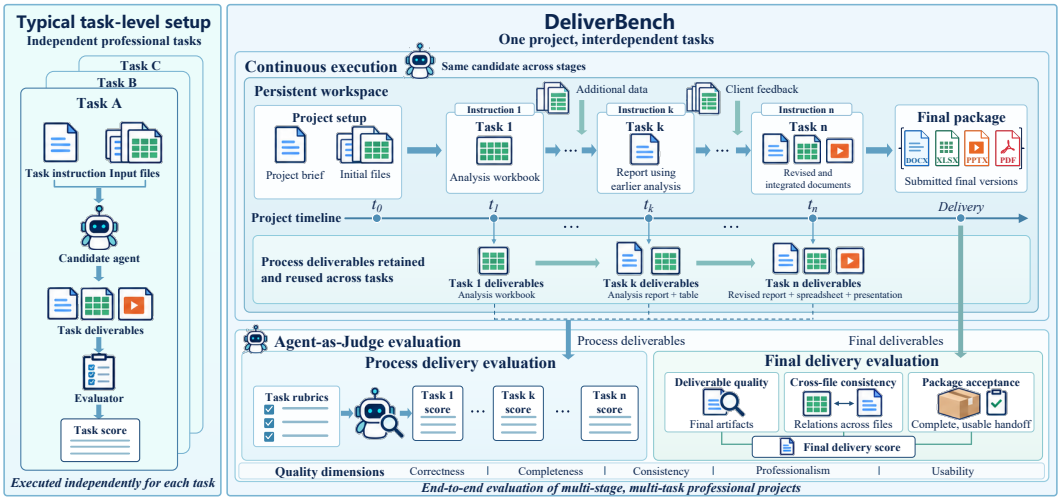


Figure 3: **Evaluation Protocol of DeliverBench.** The task-level setup executes and scores each assignment independently (left). DeliverBench runs the agent in a persistent workspace, releases new data and feedback along a project timeline, and carries intermediate artifacts into later tasks (right). Preserved checkpoint submissions and the final multi-file handoff receive separate evaluations.

Tasks share a persistent workspace along a project timeline. Execution starts from a project brief and initial files; instructions, additional data, and client feedback arrive at prescribed stages. Later tasks consume, revise, and integrate earlier outputs. Checkpoint versions are preserved for evaluation, while the final package contains the versions intended for handoff.

Evaluation follows both intermediate handoffs and final acceptance. Process evaluation assesses each subtask submission; final evaluation covers *deliverable quality*, *cross-file consistency*, and *package acceptance*. For project p with T_p subtasks, checkpoint artifacts $A_{p,t}$, and final package D_p, we report

$$P_p = \frac{1}{T_p}\sum_{t=1}^{T_p} \mathcal{S}(A_{p,t}, \mathcal{R}_{p,t}), \qquad F_p = \mathcal{S}(D_p, \mathcal{R}_p^{\text{final}}), \tag{1}$$

where $\mathcal{R}_{p,t}$ contains checkpoint criteria and $\mathcal{R}_p^{\text{final}}$ combines criteria for the three final-delivery targets. The scoring operator $\mathcal{S}$ aggregates rubric judgments with their specified weights, normalization, and acceptance gates, yielding scores in $[0, 100]$. Process and final scores remain separate. Final-artifact assessment reuses applicable task criteria; previous judgments are reused only when the relevant artifacts, criteria, and evaluation context are unchanged.

Each rubric criterion specifies an evaluation target and a quality dimension. Targets are checkpoint artifacts, final artifacts, cross-file relationships, and the complete package. Quality dimensions are correctness, completeness, consistency, professionalism, and usability. We adopt an agent-as-a-judge approach (Zhuge et al., 2024): the judge assesses criteria using task instructions and artifact evidence, supported by programmatic extraction and integrity checks. Appendix C specifies evaluation targets, evidence scopes, score aggregation, rubric reuse, and evaluator settings; Section 4.4 reports evaluator validation.

3.4 Dataset Composition and Task Characteristics

Figure 4 summarizes 11 projects, 150 subtasks, 661 input files, and 111 required final deliverables across seven domains: Finance, Medicine, Design, Data Operations, Strategy, Audit, and Law. Finance contains four projects, Medicine two, and each remaining domain one. Appendix B.2 describes their workflows and deliverables.

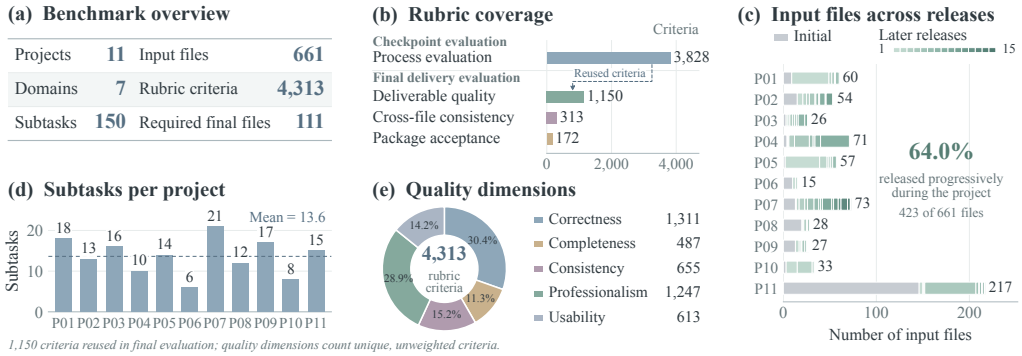


Figure 4: **Dataset overview.** (a) Benchmark scale, excluding duplicate input copies; deliverables are required final files. (b) Rubric coverage: 1,150 process criteria are reapplied to final deliverables. (c) Input releases, indexed within each project. (d) Subtask counts and their mean. (e) Unique criteria by primary quality dimension, without weighting. P01–P11 identify the same projects across panels.

Materials and workflows place different demands on agents. Finance spans transaction documents, financial workbooks, and market data, requiring numerical reconciliation and reproducible analysis. Medicine covers clinical records and follow-up in P08 and policy drafting in P09, emphasizing factual consistency and coordinated revisions, respectively. Law aligns contractual terms with changing business facts; Strategy translates financial and field evidence into recommendations; Audit links control judgments to testing evidence. Design (P04) includes 48 images among 71 inputs, requiring agents to reconcile visual materials with evolving requirements. Data Operations (P11) contains 217 inputs, mostly structured records and code-related files, emphasizing programmatic checks and version tracking.

Projects contain 6–21 subtasks (mean 13.6) and 15–217 input files. Their demands also differ in output scope: quantitative research (P06) requires 35 final files across six subtasks, including code, workbooks, reports, and figures; financial due diligence (P07) consolidates 21 subtasks into six final files and receives 15 later input releases. Overall, 423 inputs (64.0%) arrive after initialization, with 2–15 later releases per project. This share reaches 96.5% in the forensic investigation (P05) and 94.4% in Design (P04), where corrected records or client feedback require agents to revise earlier work and update dependent deliverables.

The rubric inventory contains 4,313 unique criteria: 3,828 process criteria, 313 cross-file consistency criteria, and 172 package acceptance criteria. Final deliverable-quality evaluation reapplies 1,150 process criteria to final artifact versions, counted once in the inventory. The five-dimension distribution describes annotation coverage rather than scoring weights.

4 Experiments and Analysis

4.1 Experimental Setup and Metrics

Agent systems and execution. We evaluate seven model–harness systems: GPT-5.6 Sol with Codex, Claude Opus 5 with Claude Code, Gemini 3.8 Flash with Gemini CLI, Qwen 3.8 Max with Qwen Code, DeepSeek V4.1 Flash and GLM 5.2 with Claude Code, and Kimi K3 with Kimi CLI. Projects follow the continuous protocol in Section 3.3, using Harbor 0.20.0 and Docker with a four-hour execution limit per subtask. System versions and execution settings are specified in Appendix D.1.

Evaluation and reporting. GPT-5.6 Luna judges rubric criteria; deterministic scoring produces separate process and final-delivery scores (Equation 1). We evaluate all 11 projects, yielding 77 system–project outcomes and 150 checkpoint submissions per system. Process and final-delivery scores are averaged equally across projects; the process score first averages subtasks within each project. We additionally report the fraction of checkpoint submissions scoring at least 90, pooling all 150 subtasks per system with equal weight. This thresholded subtask pass rate measures the frequency of high-scoring intermediate submissions, rather than final-package acceptance. Each system–project pair contributes one reported outcome; Appendix D.2 specifies failure and recovery handling. Appendix C.3 specifies scoring and rubric reuse; Appendix D.3 defines project means, subtask pass rates, and reporting conventions.

4.2 Main Results and Analysis

Figure 5 summarizes delivery performance, subtask pass rates, quality dimensions, and resource use across seven agent systems and 11 projects. Appendix E.1 provides the complete final-delivery and process-score matrices.

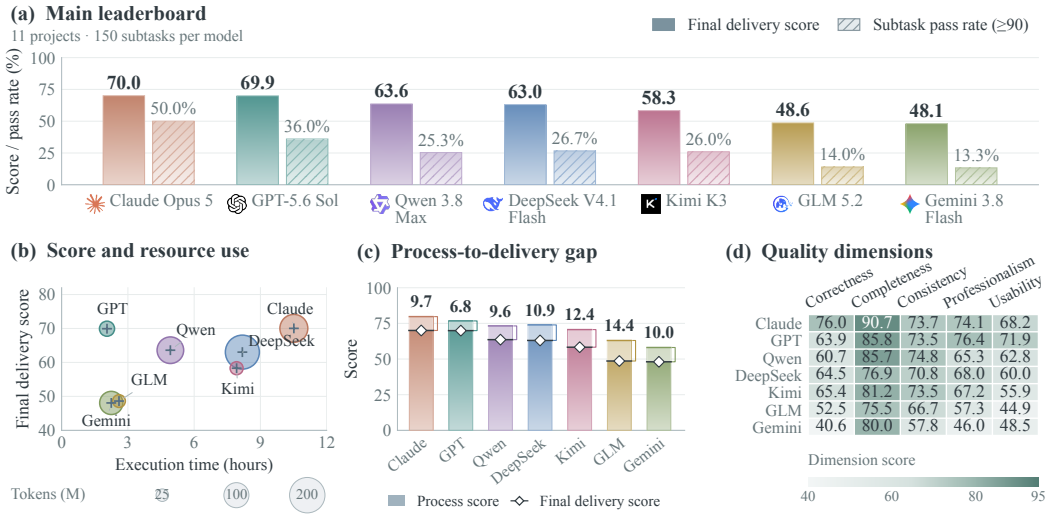


Figure 5: **Main results across seven agent systems and 11 projects.** (a) Final-delivery score and subtask pass rate ($\geq$ 90), pooled over 150 subtasks per system. (b) Execution time versus final-delivery score; bubble area encodes total tokens. (c) Process and final-delivery scores, with gaps in points. (d) Quality dimensions (color scale: 40–95), evaluated before final acceptance gates. Scores and resources use equal-weight project means.

Current agents struggle with reliable end-to-end project delivery. Final-delivery scores range from 48.1 to 70.0 out of 100, with Claude Opus 5 and GPT-5.6 Sol both reaching approximately 70 (Figure 5a). At the 90-point threshold, subtask pass rates range from 13.3% to 50.0%: Claude passes 75 of 150 subtasks, GPT passes 54 (36.0%), and each remaining system passes at most 26.7%. Even the strongest observed pass rate covers only half of the subtasks, indicating that agents do not consistently produce intermediate deliverables that meet this high-quality threshold. Project delivery

requires carrying these intermediate outputs through subsequent tasks and integrating them into a coherent final package. The low subtask pass rates and limited final-delivery scores together show that current systems fall short of sustaining high-quality delivery from intermediate work through final handoff, leaving substantial room for improvement.

Local task scores do not establish final-package readiness. Every system scores higher on process evaluation than on final delivery, with delivery gaps of 6.8–14.4 points (Figure 5c). For example, Claude scores 79.8 on intermediate work and 70.0 on final delivery, while GLM scores 63.0 and 48.6, respectively. Averaging individually scored submissions does not establish whether the final versions agree across files or jointly meet handoff requirements. Project-level evaluation checks these relationships and package conditions, assessing failure modes that local task scoring leaves untested. The gap compares different artifact versions and criteria; it does not by itself identify when or why failures arise.

Correctness and usability are the weakest quality dimensions overall. Across systems, usability averages 58.9 and correctness 60.5, below professionalism (64.9) and consistency (70.1) (Figure 5d). These scores indicate difficulties in producing accurate facts, calculations, and conclusions, and in making outputs clearly organized and operationally usable. For example, Gemini scores only 40.6 on correctness despite 80.0 on completeness, while Claude scores 68.2 on usability despite 90.7 on completeness. Completeness is the highest-scoring dimension for every system (75.5–90.7; mean 82.3), underscoring that covering requested content does not establish dependable professional delivery.

Resource costs vary widely at similar delivery quality. Motivated by joint evaluation of agent performance and cost (Kapoor et al., 2024), we examine delivery scores alongside execution time and token usage. Mean project execution time ranges from 2.05 to 10.51 hours and token usage from 26.0M to 184.5M—approximately 5.1-fold and 7.1-fold differences (Figure 5b). GPT and Claude both score approximately 70, but use 2.05 versus 10.51 hours and 36.4M versus 124.1M tokens per project. DeepSeek consumes the most tokens while scoring 63.0, showing that greater observed expenditure does not consistently accompany higher delivery scores. These comparisons characterize the evaluated model–harness pairs jointly; Appendix D.4 details resource accounting.

4.3 Case Studies: Delivery Stability and Artifact Recovery

We analyze agent execution trajectories from three projects: P01, P05, and P11. Figure 6 pairs checkpoint scores with final-delivery scores for all seven agents. Together with artifact and execution-trace inspection, this supports fine-grained analysis of how deliverable quality changes during a project, where failures arise, and whether later work repairs earlier artifacts. Appendix E.2 provides the corresponding trajectories for all 11 projects.

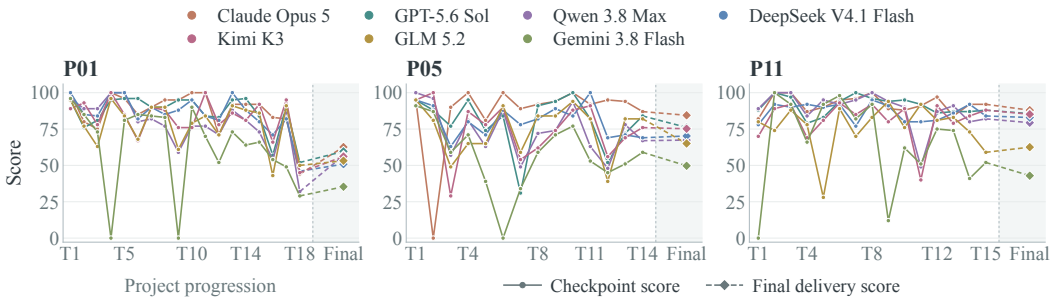


Figure 6: **Agent trajectories on three selected projects.** Circles show checkpoint scores; diamonds show final-delivery scores. Dashed terminal segments mark the change in evaluation scope. All checkpoints are retained on a common score scale. The cases illustrate quality changes and recovery behaviors during project delivery.

Final delivery builds on intermediate artifacts and introduces distinct quality requirements. The final package inherits earlier deliverables, which agents must maintain, revise, and integrate as the project unfolds. Its score is not a simple average of checkpoint scores: final evaluation also examines the resulting artifact versions, cross-file relationships, and package acceptance. The trajec-

tories reveal diverse relationships between intermediate and final quality, including late degradation in P01, recovery after an early failure in P05, and contrasting stability patterns in P11. Mean final-delivery scores nevertheless fall below mean process scores in all three cases, illustrating the delivery gap in Section 4.2. Project-delivery evaluation thus probes a capability beyond completing isolated professional tasks: maintaining accumulated work and transforming it into a coherent, usable final package.

Stronger agents generally sustain more stable delivery quality. For example, the Claude agent maintains scores of 82–100 throughout P11, whereas the Gemini agent ranges from 0 to 100 and suffers another sharp drop at T09. The Gemini agent also exhibits substantial fluctuations in P01 and P05, and obtains the lowest final-delivery score in all three cases: 35.3, 49.7, and 42.9, respectively. Its occasional high-scoring checkpoints therefore do not translate into sustained quality across the project or a high-scoring final package.

Beyond stability, stronger agents exhibit a second core delivery capability: repairing intermediate artifacts. In P05, the Claude agent scores 0 at the second checkpoint because the required transaction-reconciliation workbook was not submitted. During the third task, it detects the missing earlier deliverable and rebuilds the workbook from retained parsing results so that downstream entity-analysis references can resolve. The rebuilt workbook contains 3,327 transaction rows and is preserved in the third-task archive. The agent subsequently obtains the highest final-delivery score on P05, 84.5, compared with 49.7 for the Gemini agent. This repair restores a dependency needed downstream; the execution trace establishes recovery of an earlier artifact beyond a score rebound on another task.

4.4 Evaluation Reliability

Comparisons with human judgments provide an empirical check on model-based evaluation (Zheng et al., 2023). We assess evaluator reliability by comparing expert and evaluator judgments on final deliverables produced by GPT-5.6 Sol–Codex for three projects: P01, P05, and P10. Across 478 rubric decisions, the judgments agree on 446 (93.31%); agreement is 93.04% for deliverable quality, 95.00% for cross-file consistency, and 92.45% for package acceptance. Appendix F reports the per-project results and aggregation procedure. The check covers final-delivery judgments for this agent system, including source-task criteria applied to final artifacts; agreement for other systems or checkpoint submissions requires separate assessment.

5 Conclusion and Limitations

DeliverBench evaluates agents on multi-stage professional projects reconstructed with domain experts. It assesses intermediate submissions and final packages as agents incorporate new information and revise earlier artifacts in a persistent workspace.

Across seven model–harness systems, the highest mean final-delivery score is approximately 70 out of 100, and the highest subtask pass rate is 50% at the 90-point threshold. Every system has a lower mean final-delivery score than mean process score, with correctness and usability the weakest dimensions overall. Case studies show the importance of maintaining and repairing earlier artifacts, while greater runtime and token use do not consistently accompany better delivery. These results support evaluating final packages alongside intermediate work. More reliable project delivery could broaden enterprise delegation and support one-person companies (OPCs), where an individual sets business goals and oversees agents through final handoff.

Our study has several limitations. Coverage remains limited to eleven projects across seven domains. Prescribed information releases and feedback support reproducible evaluation but do not fully capture open-ended client communication or live business decisions. Future work can broaden project coverage, introduce richer interaction, and extend expert validation to assess delivery capability more comprehensively in professional settings.

AI USE STATEMENT

We used generative AI to draft, translate, and polish manuscript text; discover related literature; provide feedback on research methodology; and assist with data processing, analysis code, result interpretation, and figure preparation. During benchmark construction, generative AI assisted in drafting and refining a subset of rubric criteria grounded in project materials and delivery requirements. DeliverBench is grounded in real-world projects reconstructed with domain experts, with AI-assisted rubric drafting forming one part of this construction process. Domain experts reviewed the rubrics for factual support and alignment with project requirements, as described in Section 3.2. We also use an LLM-based evaluator, whose configuration and expert comparison are reported in Appendices C.4 and F. The authors take responsibility for the final text, methods, claims, and artifacts, including those developed with generative AI assistance.

ETHICS STATEMENT

All project materials used in DeliverBench were de-identified, and their use for this research was authorized by the contributing domain experts. Research use and any sharing of these materials remain subject to the applicable permissions and confidentiality obligations. The benchmark evaluates professional work in project workspaces; agent-generated financial, legal, medical, and other deliverables are research artifacts and do not constitute professional advice or authorization for real-world action. The reported scores characterize performance on the benchmark and do not establish suitability for unsupervised deployment in professional practice.

REPRODUCIBILITY STATEMENT

Sections 3.2 and 3.3 describe benchmark construction and the execution and evaluation protocol. Appendix B provides a representative workflow and project profiles. Appendices C and D document evaluation targets, aggregation rules, rubric reuse, evaluator settings, agent configurations, recovery handling, and resource accounting. Appendix E reports complete per-project scores and trajectories, and Appendix F reports expert–evaluator agreement. These descriptions support inspection of the experimental setup and reported results.

REFERENCES

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault L De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *Advances in Neural Information Processing Systems*, 37:5996–6051, 2024.

Sayash Kapoor, Benedikt Stroebl, Zachary S Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. *arXiv preprint arXiv:2407.01502*, 2024.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

Thomas W Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)*, 26(1):87–119, 1994.

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.

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.

Summer Sun. Sqbench: A benchmark for evaluating task delivery by language-model agents in production-oriented workflows. *arXiv preprint arXiv:2607.23123*, 2026.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, Haoyun Zhang, Jeffrey Lin, et al. Agents' last exam. *arXiv preprint arXiv:2606.05405*, 2026.

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.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, et al. Apex-agents. *arXiv preprint arXiv:2601.14242*, 2026.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. Officebench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*, 2024.

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, 2026.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Haoran Zhang, Luxin Xu, Zhilin Wang, Runquan Gui, Shunkai Zhang, Haodi Lei, Zihao He, Bingsu He, Chicheng Qin, Tong Zhu, et al. π-Bench: Evaluating proactive personal assistant agents in long-horizon workflows. *arXiv preprint arXiv:2605.14678*, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Liya Zhu, Xin Ma, Tao Liu, Haodong Wang, Ge Zhang, Jingzhe Ding, Qingshui Gu, Yongjie Zhong, Jinxiang Meng, Yuan Gao, et al. Startupbench: Benchmarking general-purpose agents on market-validated end-to-end workflows. *arXiv preprint arXiv:2608.17800*, 2026.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, et al. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*, 2024.

A Benchmark Comparison Details

Table 1 compares documented benchmark designs, not model performance or the fraction of instances with each feature. A checkmark denotes explicit support under the definitions below, a triangle denotes limited or related support, and a cross denotes support not specified in the cited protocol; it is not a claim that the underlying agent could never perform the behavior.

Evolving inputs/feedback. Full support requires new task-relevant materials, requirements, or substantive collaborator feedback during execution that guide subsequent work. Scheduled releases and human or simulated-colleague responses qualify; ordinary tool observations, errors, generic continuation prompts, and retrieval of pre-existing content do not. TheAgentCompany describes incorporating a simulated manager's feedback (Xu et al., 2026, Section 3 and Appendix C); DeliverBench schedules materials and client feedback across stages (Section 3.3). Other protocols do not specify this mechanism: GDPval is one-shot (Patwardhan et al., 2026); APEX-Agents is single-turn without clarification (Vidgen et al., 2026); OfficeBench returns operation observations (Wang et al., 2024); and WorkArena++ provides initial instructions or tickets (Boisvert et al., 2024). PaperBench supplies author clarifications at the outset (Starace et al., 2025); StartupBench and Workspace-Bench initialize task workspaces (Zhu et al., 2026; Tang et al., 2026); SQBench specifies fixed inputs (Sun, 2026).

Long-horizon tasks. Long-horizon support follows an explicit benchmark objective or documented substantial compositional execution, without imposing a common numerical step threshold. SQBench receives partial support because its suite mixes atomic capabilities with composite and domain workflows; a checkmark elsewhere does not imply that every instance is long.

Explicit subtasks and artifact dependencies. Full subtask support requires instantiated component tasks within a workflow or project. Steps, checkpoints, or hierarchical requirements within one assignment receive partial support, including PaperBench's hidden rubric (Starace et al., 2025) and DevAI's requirement DAG (Zhuge et al., 2024). Full cross-task artifact support requires an explicit handoff of generated outputs between separately specified component tasks. Within-workflow artifact or application-state dependencies receive partial support: examples include OfficeBench's spreadsheet-to-email workflow (Wang et al., 2024), WorkArena++'s user-to-hardware assignment, TheAgentCompany's build-to-launch sequence (Xu et al., 2026), and DevAI's script-to-image requirements. Relationships among supplied input files alone do not establish generated-output handoffs. In particular, Workspace-Bench restores modified files after each independent task (Tang et al., 2026).

Intermediate evaluation. Full support means that explicitly defined intermediate subtasks or milestones receive checks or scores, as in WorkArena++, TheAgentCompany, and DevAI. This does not require online grading or preserved chronological snapshots: TheAgentCompany and DevAI can inspect state and trajectories after execution. Partial support covers partial-progress assessment from a final submission, or general retrospective process/behavior checks, as in PaperBench, Workspace-Bench, and SQBench. Neither type alone establishes separate quality measurements of preserved stage submissions along the original execution timeline.

Project-level delivery. Full support means evaluating an integrated multi-file project submission against a shared project outcome. This includes PaperBench's full repository reproduction protocol, not just professional Office-file packages; the comparison uses full PaperBench rather than its Code-Dev variant (Starace et al., 2025). Task- or workflow-level deliverable checks receive partial support when a project-package acceptance protocol is not specified. This gives credit to StartupBench's rubric evaluation of a set of deliverables (Zhu et al., 2026), and to final-output checks in the other task benchmarks, without equating them to a project acceptance stage. WorkArena++ instead validates application task state.

Verification scope. The comparison was checked against the cited primary papers and available public implementations. SQBench's official tasks and production scorer are withheld; its entries therefore describe the published protocol, not an independent audit of hidden tasks (Sun, 2026). The DeliverBench row describes the protocol in Section 3.

B Project Example and Profiles

The benchmark comprises eleven projects spanning strategic advisory, transactions, audit, design, investigation, quantitative research, healthcare, legal services, and data delivery. Each project unfolds through a sequence of business assignments in which new materials and feedback must be reconciled with earlier work. Projects are identified exclusively as P01–P11 throughout the paper; the descriptions below omit organization and individual names. They describe the benchmark tasks, not the performance of a particular model.

B.1 Representative Project Workflow

P05 illustrates how a long project combines evidence acquisition, analysis, review, revision, and final delivery. Figure 7 organizes its fourteen subtasks (T01–T14) into four presentation phases. Each subtask specifies incoming materials, the work requested, its deliverables, and the information carried into later work. The final handover comprises six main files plus supporting evidence. The example combines multiple file formats, delayed evidence, reviewer-driven revisions, constrained information acquisition, and a traceable final delivery.

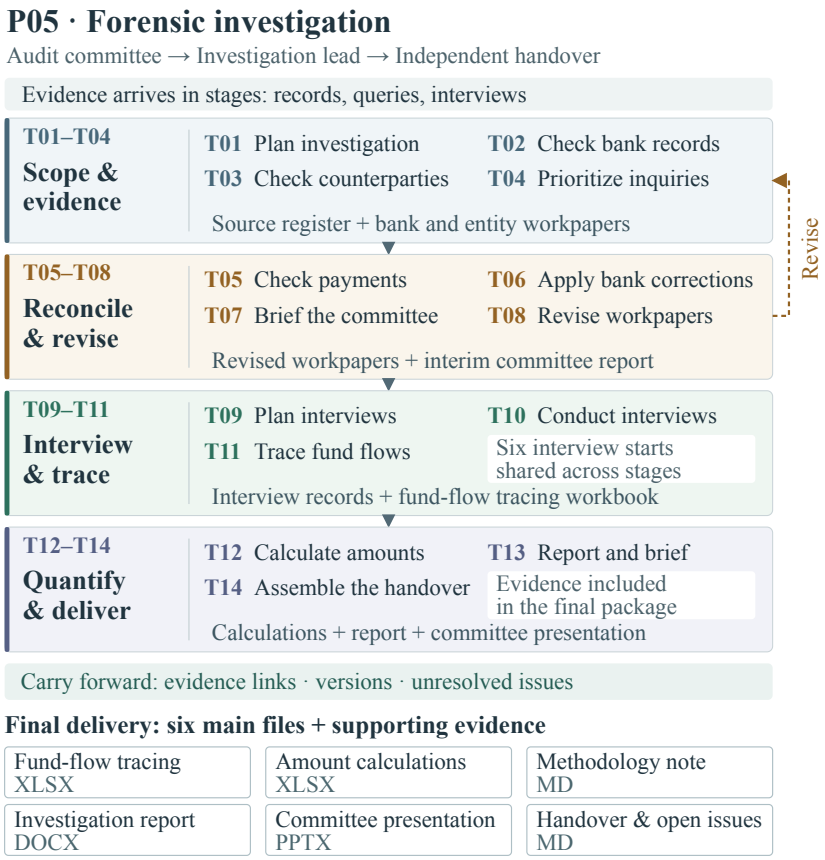


Figure 7: Required workflow for P05. Four phases group fourteen subtasks (T01–T14). Corrected bank records and review comments trigger revisions to earlier workpapers, as indicated by the dashed return arrow; interviews and later account records inform fund-flow tracing and amount calculations. Final delivery combines editable conclusions, workbooks, supporting evidence, and unresolved-issue records. The phase grouping summarizes the task requirements rather than prescribing an additional execution protocol or depicting a particular model run.

B.2 PROJECT PROFILES

P01. The agent acts as an external strategy and capital adviser to a manufacturing company, supporting management and board decisions on operating improvement and industrial cooperation. The project begins with a source register and a reproducible financial baseline, then incorporates site visits, prospective partner profiles, industry comparisons, and alternative cooperation structures. New materials and review comments require the agent to update both the underlying analysis and its recommendations. The final delivery combines an analytical workbook, strategy and cooperation proposals, an investment decision memorandum, a management presentation, and handover notes, with assumptions, evidence gaps, and decisions requiring approval kept explicit.

P02. The agent participates in a transaction team preparing the syndicated refinancing of a digital infrastructure business. Financial, environmental, engineering, and tax due diligence feed into evolving financing terms, cash-flow models, committee materials, and lender coordination. As new evidence and lender responses arrive, the agent must reconcile assumptions, financing amounts, valuation references, and the status of commitments across documents. The final package contains a scenario model, an assumptions memorandum, a syndication update, a valuation presentation, and draft signing documents. The task ends at signing readiness; proposed terms, conditional commitments, and completed approvals must remain distinguishable.

P03. The agent manages an external audit of information technology general controls to support a financial audit team's reliance assessment. Work progresses from project planning and system understanding to scope definition, a risk-control matrix, evidence requests, walkthroughs, and tests across six control domains. The agent must track missing evidence, reconcile scope and control status across workpapers, and respond to cross-domain review findings without treating incomplete tests as effective controls. Final deliverables include the audit workpapers, an issue register, conclusions on reliance and financial-audit implications, and a management presentation with outstanding actions and review status.

P04. The agent serves as a design project manager reviewing and coordinating a residential interior design project. Successive client discussions and design submissions change room functions, stylistic preferences, furniture choices, cabinetry, lighting, and equipment interfaces. The agent maintains requirements, decisions, and issue registers, checks the implications of each revision, and resolves inconsistencies between drawings and written recommendations. The final delivery assembles a design review report, meeting presentation, acceptance documents, and the relevant supplied drawings and visual materials. The task concerns reviewing and coordinating an evolving design, rather than independently redrawing all original design attachments.

P05. The agent leads a forensic consulting engagement commissioned by an audit committee to investigate supply-chain fund flows. Across fourteen stages, it establishes the investigation scope, reconciles bank and business records, checks counterparty identities, prioritizes inquiries, and revises working papers when corrected records and review comments arrive. It then plans and conducts interviews through the provided tools, integrates additional account evidence, traces fund movements, and develops reproducible amount calculations. The final package contains tracing and calculation workbooks, a methodology note, an editable investigation report, a committee presentation, and handover notes, supported by locally included evidence and unresolved-issue records.

P06. The agent acts as a quantitative researcher assessing an equity-index futures research program and its readiness for simulated observation. Six stages cover data auditing, candidate factors, validation and selection, a frozen modeling plan, execution and account-state simulation, and independent evaluation. The work must connect time-consistent research decisions to implementable trades, reproducible performance calculations, and explicit account constraints. Final deliverables include data and research workbooks, executable code, reports, figures, and a documented recommendation for the next research stage. A supported decision not to trade or not to proceed is permissible; positive returns are not a prerequisite.

P07. The agent conducts investment-side financial due diligence on a manufacturing group. It establishes the entity and consolidation perimeter, reconciles a multi-period financial baseline, and develops workpapers on revenue, customers, production costs, margins, inventory, working capital, debt, capital expenditure, and personnel costs. Management responses and supplementary materials must be incorporated into the relevant analyses, earnings adjustments, and forecast assessment. Final deliverables comprise a due diligence report, a supporting databook, an earnings-quality adjustment

schedule, an open-issues register, a management presentation, and handover notes. Findings must connect to valuation, funding needs, and transaction decisions while retaining evidence limitations.

P08. The agent processes a post-marketing individual case safety report from initial receipt through closure. It registers the report, constructs a source-linked clinical fact table, identifies follow-up needs, and integrates later responses into successive case versions. Medical assessment and reporting classification inform the case narrative and structured safety-report fields, which are subsequently revised in response to independent quality-control comments. The agent then reconciles review decisions and submission receipts before preparing the final narrative, field workbook, closure summary, and archive inventory. Clinical facts, administrative status, source provenance, and unresolved questions must remain consistent throughout the case lifecycle.

P09. The agent advises a pharmaceutical company developing its commercialization compliance framework. Initial document review and departmental interviews establish business facts, risk priorities, and governance recommendations. Successive stages draft and revise policies for healthcare-professional interactions, material review and publication, third-party due diligence, conflicts of interest, investigation cooperation, and business conduct. Cross-functional feedback must be propagated across policy text, approval procedures, forms, and implementation tools. The final delivery brings together twelve policy and supporting files, a management presentation, and handover recommendations, preserving the distinction between existing controls, proposed changes, unresolved choices, and actions that have actually been approved.

P10. The agent acts as external legal counsel to a platform connecting business clients with professional contributors. It reviews the business model and data flows, updates the user agreement, and drafts a privacy policy and data-submission agreement. Client comments and an additional cross-border pilot require coordinated revisions across the three documents and the proposed launch arrangements. Final deliverables include the publishable legal documents, a launch checklist, and handover notes identifying unresolved decisions and release conditions. The documents must reflect the platform's actual operating and data-handling arrangements without implying that proposed activities or approvals are already in place.

P11. The agent coordinates the acceptance and release of an evaluation-data delivery assembled from multiple production streams. Incoming task packages, model responses, scoring records, and client feedback must be registered, deduplicated, repaired, and reconciled across versions. The project includes runtime integration, evidence review, targeted testing within a constrained window, and the handling of replacements and supplementary deliveries. Final outputs comprise the adopted customer package and its supporting evidence, manifests, delivery notes, and internal registers for feedback, status, and remaining work. Production claims, verified results, approvals, dispatch, and client acceptance must be recorded as distinct states.

C Evaluation Protocol and Scoring

C.1 Evaluation Targets and Quality Dimensions

Process delivery evaluation assesses each subtask's checkpoint artifacts under the information available at submission. Final delivery evaluation uses the three targets in Table 2. The evaluation target identifies the scope of assessment; the quality dimension in Table 3 identifies the property being checked. Each criterion has one primary quality dimension. Checks that combine independently judgeable properties should be separated before scoring. The examples below illustrate the categories rather than report observed model failures.

Table 2: Final-delivery evaluation targets and their scopes.

Target	Scope	Example criterion
Deliverable quality	Individual deliverables in their submitted final versions.	The final report incorporates the revisions required by the end of the project.
Cross-file consistency	Agreement between two or more final artifacts.	The report and workbook agree on the operative assumptions, units, and totals.
Package acceptance	The complete file collection against the project's shared delivery requirements.	All required deliverables are included and their handoff status is clearly communicated.

Table 3: Quality dimensions. The same dimension can apply to different evaluation targets.

Dimension	Assessed property	Illustrative failure
Correctness	Accuracy of facts, calculations, reasoning, and conclusions under the applicable information conditions.	A conclusion contradicts the supporting data.
Completeness	Coverage of required content, files, and project objectives.	A required analysis or deliverable is missing.
Consistency	Agreement of figures, definitions, decisions, and version states within and across artifacts.	A revised assumption is not reflected in an affected report.
Professionalism	Adherence to domain methods, terminology, business rules, and responsibility boundaries.	A provisional finding is presented as an authorized final decision.
Usability	Organization, presentation, and operational accessibility for the intended recipient.	An otherwise valid workbook cannot be interpreted or used for its intended decision.

Usability includes readability and delivery readiness when required by the task. The three final-delivery targets describe assessment scope; they need not correspond to three separately authored rubric files. In particular, deliverable-quality assessment can reuse the relevant source-task criteria, applied to the final versions.

C.2 Criterion Evidence and Scope

Criteria identify the applicable task requirement, target artifacts, scoring weight, and evidence needed for judgment. A consistency check should identify the relevant files, fields, or decisions rather than merely require that the package be "consistent." Evaluator evidence and agent-facing output requirements serve different purposes: evaluators need a basis for verification, while an artifact is required to display citations or provenance only when its task or professional standard calls for them. Evidence traceability is therefore a task-specific requirement, not an additional universal quality dimension.

Process judgments apply to the artifact version and information conditions at the checkpoint. Final judgments apply to the submitted package and the requirements in effect at project completion. Reusing a prior judgment requires unchanged artifacts, applicable criteria, and evaluation context;

file identity alone is insufficient when a criterion depends on other package contents. Scores for process delivery and final delivery are reported separately, and the same requirement must not receive duplicate weight within a single aggregate merely because it is checked at multiple points.

C.3 Score Aggregation and Rubric Reuse

Rubric weights determine the contribution of each requirement. The operator $\mathcal{S}$ in Equation 1 applies the scoring specification associated with its rubric set. Positive criteria award credit, negative criteria deduct credit when their specified failure condition is met, and dependencies are resolved according to the criterion's declared scoring rule. Scores are normalized and bounded to $[0, 100]$. Mandatory acceptance criteria act as gates: a triggered fatal condition sets the affected score to zero, including the final score when the condition applies to final delivery. Criterion judgments, ungated component scores, and gate outcomes are retained separately for diagnosis.

Final delivery combines three complementary sets of criteria. Write $\mathcal{R}_p^{\text{final}} = \mathcal{R}_p^D \cup \mathcal{R}_p^C \cup \mathcal{R}_p^A$, corresponding to deliverable quality, cross-file consistency, and package acceptance. Their contributions follow the weights and normalization specified in the project's scoring configuration. The three target labels describe what is assessed; they do not imply a fixed benchmark-wide allocation of weight. The process aggregate P_p is not added to F_p.

For the reported results, let $d_{m,p}$, $c_{m,p}$, and $a_{m,p}$ denote the three normalized component scores after their applicable unit-level rules. Let $g_{m,p} = 1$ denote a triggered project-level fatal gate. The final score is

$$F_{m,p} = \begin{cases} 0, & g_{m,p} = 1, \\ \alpha_p d_{m,p} + \beta_p c_{m,p} + \gamma_p a_{m,p}, & g_{m,p} = 0. \end{cases} \tag{2}$$

The component coefficients sum to one for every project and are identical across systems within a project. The process score is not a component of this final score.

Package acceptance is a rubric score, not an acceptance rate; any binary success metric additionally requires an explicit decision rule.

Reused criteria retain their source identity. Final deliverables are mapped to their originating subtask criteria, which are applied to the submitted final versions. When several files share a source-task rubric, that source group is included once rather than repeated for every file. A checkpoint verdict is reused only if the relevant artifact versions, requirements, and evidence context remain unchanged; otherwise the applicable criteria are judged again. Criterion reuse therefore avoids duplicate authoring and counting, while still allowing final revisions to be reassessed. The unique-criterion inventory in Figure 4 counts a reused criterion once, even when it is evaluated at both a checkpoint and final handoff.

Quality dimensions provide a separate diagnostic view. For each dimension, we normalize the weighted credit of its applicable criteria by their available positive weight, retaining the scoring specification's treatment of penalties and dependencies. For Figure 5d, the applicable criteria are those selected by the current final-delivery configuration, weighted by their project-level coefficients. Signed awards include dependency resolution and penalties; normalization by positive available weight and clipping to $[0, 100]$ take place within each project and dimension, before averaging equally across the 11 projects. The profiles precede unit-level clipping and final acceptance gates, so they are diagnostic views rather than a five-part decomposition of the final score. Dimensions without applicable criteria are marked as unavailable. These profiles describe criterion-level performance; a final acceptance gate can invalidate a package even when some dimension or component scores remain high.

C.4 Evaluator Configuration

We use GPT-5.6 Luna with high reasoning effort. Criteria are evaluated in batches of eight with up to four concurrent batches. Each batch has at most two attempts and a 1,200-second request timeout; unresolved items receive a single-item fallback attempt. Retries repair missing evaluator responses rather than select the most favorable judgment. The verifier has a separate 43,200-second timeout and is not charged against the agent's subtask execution budget. The judge evaluates criterion evidence; programmatic scoring handles weights, dependencies, normalization, and acceptance gates.

Evaluator instructions, private references, and verdicts are kept outside the agent-facing workspace; scheduled client feedback is part of the project input.

D EXPERIMENTAL DETAILS

D.1 AGENT SYSTEMS AND EXECUTION CONFIGURATION

Each evaluated system consists of a model and its execution harness. Table 4 lists the model–harness pairs and CLI versions recorded in the evaluated configurations. Harnesses retain their own tool interfaces and context-management behavior, so comparisons concern complete agent systems.

Table 4: Agent-system configurations and CLI versions.

Model	Harness	CLI version
GPT-5.6 Sol	Codex	0.142.4
Claude Opus 5	Claude Code	2.1.278
Gemini 3.8 Flash	Gemini CLI	0.60.0
Qwen 3.8 Max	Qwen Code	0.24.1
DeepSeek V4.1 Flash	Claude Code	2.1.278
GLM 5.2	Claude Code	2.1.278
Kimi K3	Kimi CLI	1.50.0

Projects run continuously in a persistent workspace. Harbor 0.20.0 orchestrates Docker environments with session continuation enabled. Each project provides an initial brief and files, then releases subtask instructions and additional materials at the prescribed points in its timeline. The agent carries its own files and conversation state into subsequent work; harness-managed context compression may occur. Checkpoint artifacts are preserved, and final delivery is evaluated on the versions designated for handoff. Project-specific environments provide the file-processing tools and business utilities needed by their tasks. Future-stage inputs and evaluator-only material remain unavailable until their intended release, or remain private throughout evaluation, respectively.

Execution limits are specified per subtask. The configured agent limit is 14,400 seconds per subtask, with a separate 1,200-second agent-setup limit. The standard project container requests two CPUs, 8 GiB RAM, and 30 GiB storage, with network access enabled. The common configuration requests high reasoning effort where the harness and endpoint expose that setting; this does not impose an identical effective reasoning budget across models. No common token cap or temperature is imposed by the launcher. Judge settings and timeouts are specified separately in Appendix C.4.

D.2 RUN PROVENANCE AND FAILURE HANDLING

Each system–project pair contributes one reported outcome. Job configurations request one attempt and disable automatic trial retries. Infrastructure recovery may nevertheless create additional attempt records or resume from a preserved checkpoint. These records distinguish new agent generation, reused checkpoint artifacts, and evaluator-only recovery; multiple attempts are not treated as independent repetitions. When a successful agent submission exists but judging or storage fails, recovery preserves that submission rather than regenerating it to obtain a different score.

Execution status is distinct from artifact quality. A missing required deliverable in a completed submission is evaluated by the applicable criteria and acceptance gates. Pending runs, interrupted infrastructure, and missing judge results retain explicit status and are not silently converted into observed quality-zero scores. Completed evaluations that trigger a fatal delivery condition retain their zero final score and underlying evidence. The reported resource totals cover the selected scored stages and retained sessions; they do not measure the full overhead of all failed or recovered attempts.

Compatibility changes remain part of the system configuration. For the evaluated GLM system, native PDF and image payloads generated by Claude Code are unsupported. The recorded compatibility guard returns a recoverable tool error for these unsupported requests while leaving local files and extraction tools available. Such adaptations and checkpoint recoveries belong in the run provenance because they affect the execution interface. The tables in Appendix E.1 cover the 77 selected system–project outcomes. Their process means use 1,050 checkpoint evaluations from the retained score export. Historical results with different scoring configurations are not pooled without an explicitly documented reassessment.

D.3 Aggregation and Reporting

Projects receive equal weight in cross-system comparisons. Within-project scores follow Equation 1 and Appendix C.3. For a shared set $\mathcal{P}$ of evaluated projects, the aggregate for system m is

$$\overline{F}_m = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} F_{m,p}, \qquad \overline{P}_m = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} P_{m,p}. \tag{3}$$

The project set and completion coverage accompany each comparison.

Subtask pass rate and delivery gap. For each system, the pass rate at 90 is the number of historical checkpoint scores at least 90 divided by 150, including the original acceptance gates and comparing unrounded scores. Final-source rubric reuse does not create additional subtasks in this denominator. This is a subtask-weighted statistic, whereas the process and final means in Equation 3 give equal weight to projects. Figure 5c reports $\overline{P}_m - \overline{F}_m$ as a descriptive difference between these means; their different evaluation scopes prevent interpreting it as a calibrated capability loss.

Missing outcomes are shown explicitly, and averages over different available project subsets are not presented as directly comparable rankings.

D.4 Resource Accounting

Runtime. Agent execution and evaluation overhead are reported separately. Agent runtime is measured from the per-subtask execution intervals and summed over the project; it includes model waiting time and tool execution within those intervals. Environment setup, judging, and recovery overhead are tracked separately. For recovered runs, retained stages use their original execution records rather than the outer recovery job's start and finish times. Additional work from failed attempts is identified separately from the execution that produced the reported artifacts.

Tokens and cost. Native session records distinguish input, output, and cached-token fields; unavailable values are preserved as missing. Some harnesses report cumulative usage after each resumed subtask, so summing these snapshots would count earlier requests repeatedly. Project totals require deduplication by session and request identity; cumulative differences are valid only within a confirmed continuous session. Cache reads are not added again when they are already included in input usage. Provider differences in tokenization, reasoning-token reporting, and caching are documented alongside efficiency comparisons. A recorded or estimated cost is distinguished from a verified API charge, and absent cost fields are not interpreted as zero.

For Figure 5b, execution times cover all 77 reported system–project outcomes and all 1,050 selected subtasks. Of these subtasks, 1,047 use host-recorded agent intervals, two use native CLI durations, and one uses an interval inferred from native session timestamps. Token totals are deduplicated from retained native records, including identifiable logged subagents, and are available for all 77 outcomes. Total tokens equal input plus output; cached input is a subset of input and is not added again. The plotted resources cover the selected scored stages and retained sessions, not the full cost of every historical failed attempt or recovery operation. Token counts are usage measures rather than comparable monetary charges across providers.

E Complete Project Results

E.1 Agent–Project Scores

Tables 5 and 6 report all 77 system–project pairs for final delivery and process evaluation, respectively. Scores range from 0 to 100 and are rounded to two decimals only for display. Each column identifies the corresponding model–harness system in Table 4; P01–P11 follow Appendix B.2. The mean row gives equal weight to the eleven projects. Final scores include their applicable acceptance gates; process scores average the historical checkpoint scores within each project. These are different evaluation targets, as specified in Section 3.3.

Table 5: Final-delivery scores for all seven agent systems and eleven projects.

Project	Claude	GPT	Qwen	DeepSeek	Kimi	GLM	Gemini
P01	62.62	59.62	56.77	50.86	56.11	53.21	35.32
P02	57.11	45.89	50.49	55.05	0.00	54.61	52.61
P03	71.40	71.18	67.59	69.12	68.60	57.29	35.21
P04	58.67	73.95	67.60	73.66	59.06	64.23	55.65
P05	84.46	76.13	67.41	70.24	75.19	65.07	49.73
P06	70.14	69.82	67.97	57.92	59.80	47.48	40.45
P07	56.84	54.53	53.02	48.94	53.46	35.75	49.58
P08	74.28	71.84	63.64	70.16	74.00	0.00	57.40
P09	66.65	78.49	61.93	52.34	54.23	49.12	56.47
P10	80.23	82.23	63.79	62.19	55.67	45.55	53.44
P11	87.97	85.59	79.37	82.93	85.17	62.54	42.93
Mean	**70.03**	**69.93**	**63.60**	**63.04**	**58.30**	**48.62**	**48.07**

Table 6: Mean process scores within each project, based on its checkpoint submissions.

Project	Claude	GPT	Qwen	DeepSeek	Kimi	GLM	Gemini
P01	87.33	86.17	78.39	85.61	82.50	78.17	63.00
P02	50.69	35.15	44.54	43.00	44.31	37.46	44.15
P03	82.00	78.88	84.00	80.44	80.75	76.81	64.19
P04	76.60	86.20	82.20	87.20	82.80	81.50	80.80
P05	86.36	79.93	73.57	80.71	74.93	75.14	56.86
P06	75.67	75.54	70.83	62.33	62.25	55.71	44.88
P07	88.42	83.39	80.30	78.42	79.27	57.52	63.82
P08	79.83	72.92	65.83	68.50	61.00	51.42	50.58
P09	85.63	79.71	71.13	78.18	72.56	59.63	54.99
P10	72.56	75.22	66.12	62.28	55.38	43.16	49.88
P11	92.20	91.07	88.53	86.67	81.47	76.53	65.93
Mean	**79.75**	**76.74**	**73.22**	**73.94**	**70.66**	**63.00**	**58.10**

E.2 PROJECT TRAJECTORIES

Figure 8 extends the three cases in Section 4.3 to all 11 projects and seven agents, retaining all 1,050 checkpoint scores and 77 final-delivery scores. The panels use the same score scale and agent colors, allowing the quality changes discussed in the main text to be examined across the full benchmark.

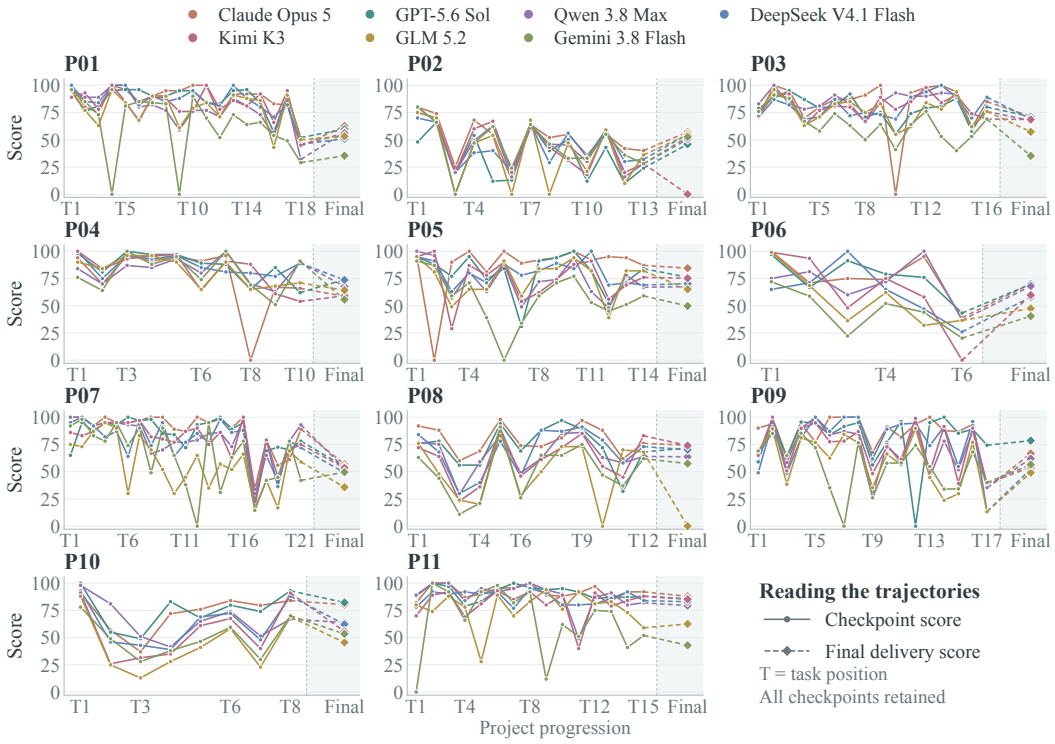


Figure 8: **Agent trajectories across all 11 projects.** Panels are ordered P01–P11. Circles show checkpoint scores; diamonds show final-delivery scores. T labels denote chronological task positions, with all original checkpoints retained. Dashed terminal segments mark the change in evaluation scope. All panels share the same score scale and agent colors.

F EVALUATOR VALIDATION

Domain experts checked the final-delivery evaluations of the GPT-5.6 Sol–Codex experimental system on P01, P05, and P10, covering 478 rubric decisions. These comparisons concern the final artifacts produced by this system, not a pooled sample from all seven systems. We compare their binary judgments with the evaluator's judgments for deliverable quality, cross-file consistency, and package acceptance. Table 7 reports agreement for each project and evaluation target, together with pooled results.

Table 7: Expert–evaluator agreement for final deliverables produced by GPT-5.6 Sol–Codex. N is the number of compared criteria. Entries give matching judgments/compared criteria and the corresponding percentage.

Project	N	**Overall**	**Deliverable quality**	**Cross-file consistency**	**Package acceptance**
P01	151	141/151 (93.38%)	94/101 (93.07%)	29/30 (96.67%)	18/20 (90.00%)
P05	133	124/133 (93.23%)	85/91 (93.41%)	22/24 (91.67%)	17/18 (94.44%)
P10	194	181/194 (93.30%)	142/153 (92.81%)	25/26 (96.15%)	14/15 (93.33%)
Pooled	**478**	**446/478 (93.31%)**	**321/345 (93.04%)**	**76/80 (95.00%)**	**49/53 (92.45%)**

Agreement is the number of matching binary judgments divided by the number of compared criteria. Pooled rates are computed by summing matching judgments and compared criteria across projects, without applying delivery-score weights. The three evaluation targets account for 345, 80, and 53 comparisons, respectively, with 321, 76, and 49 matching judgments. Overall agreement is 446/478 (93.31%), and per-project agreement ranges from 93.23% to 93.38%. These rates measure agreement between expert and evaluator judgments on the reviewed GPT-5.6 Sol–Codex final deliveries; they do not estimate agreement for the other six agent systems.

The reviewed final-delivery criteria include source-task rubrics applied to the final artifacts. This check assesses agreement at final handoff; assessing checkpoint agreement requires a separate comparison on the earlier artifact versions and their available information.