COSWE: EVALUATING INTERACTIVE PLANNING IN COLLABORATIVE SOFTWARE ENGINEERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents increasingly handle repository exploration, implementation, and testing, but real development often begins with only a partial description of what the user wants. Early benchmarks assessed code against predefined specifications, while recent work has brought requirement clarification and user feedback into the evaluation. This shift calls for measuring interactive planning ability: the speed and quality with which agents identify and formulate the task's target technical solution through interaction. The target reflects the configured user's requirements and preferences, without assuming a universally optimal design. We introduce PlanScore as the primary metric, combining discovery timing, resolution delay, and completion to track progress toward accepted decisions. To support this evaluation, we build COSWE, a benchmark based on a dataset of real repository changes. Each task pairs an underspecified request with private reference requirements and a stateful user simulator, so agents must clarify choices, provide behavioral evidence, and explain technical trade-offs as they develop a plan. Report fidelity complements the interaction-based score by assessing whether final reports preserve the reference requirements. Together, the dataset, interaction protocol, and evaluation framework make both the process and outcome of planning observable. Experiments reveal differences in agents' discovery and resolution performance and show that report omissions consistently exceed contradictions.

1 INTRODUCTION

Software engineering often begins with a goal rather than a complete specification. Compatibility, boundary behavior, and acceptable trade-offs may only become clear through discussion, making requirements elicitation and shared understanding integral to development (Zowghi & Coulin, 2005; Clark & Brennan, 1991). In Figure 1, a request to move deleted files to trash leaves the implementation open. The agent compares alternatives and clarifies the user's selected design, gradually turning the initial goal into a technical plan.

Coding benchmarks have progressively expanded the scope of evaluation. HumanEval and MBPP assess generated programs against predefined tasks and tests (Chen et al., 2021; Austin et al., 2021), while SWE-bench and Terminal-Bench evaluate solutions in repository and terminal environments (Jimenez et al., 2023; Merrill et al., 2026). These benchmarks establish executable correctness as a central outcome, but leave open how to evaluate an agent's ability to establish the intended solution when the initial request is incomplete.

Recent benchmarks bring planning and user interaction more directly into this picture. SWE-CI includes an Architect–Programmer loop and an explicit planning step, assessing maintainability through evolving test outcomes (Chen et al., 2026). ICAE-Bench grounds clarification in authored hidden requirements and evaluates requirement coverage, repository correctness, and design quality (Peng et al., 2026). SWE-Interact uses a persona-conditioned user to inspect the workspace, critique plans and implementations, and progressively reveal requirements; it also scores saved plans and implementation checkpoints against sub-goal rubrics (Raghavendra et al., 2026).

These advances make it possible to examine how a plan takes shape during collaboration. An agent may identify a missing decision early yet struggle to resolve it, or reach acceptance quickly and

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.

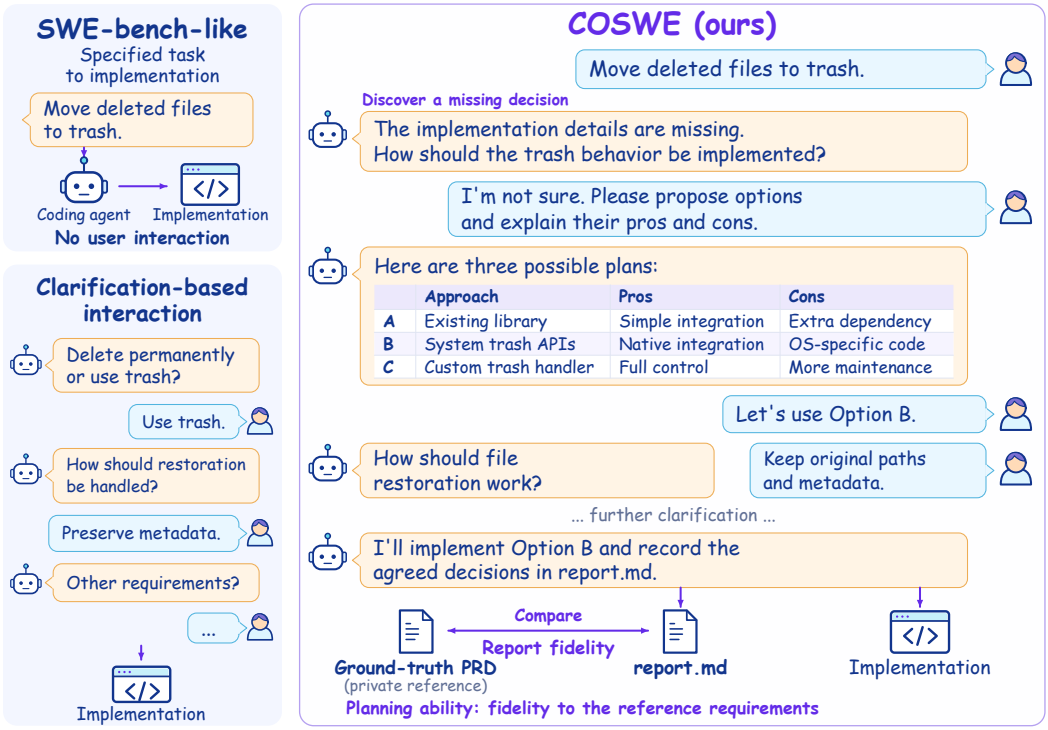

Figure 1: **Interactive planning from an underspecified request.** The agent discovers a missing decision, compares alternatives, and clarifies the selected plan. Report fidelity compares the final `report.md` with the private ground-truth PRD; PlanScore scores the interaction trace. The schematic dialogue illustrates acceptance consistent with a fixed reference, not an evaluated outcome.

then omit the decision from its final report. Such cases motivate our focus on **interactive planning ability**: the speed and quality of identifying and formulating the task's target technical solution through interaction. This target reflects the configured user's requirements and preferences, without claiming universal technical optimality.

To measure this ability, we distinguish discovering a missing decision from reaching an accepted resolution. Our primary metric, **PlanScore**, combines discovery timing, resolution delay, and completion, retaining unresolved facets as zeros. **Report fidelity** complements this process measure by assessing whether final reports preserve reference requirements.

We introduce **COSWE**, a benchmark with tasks derived from real repository changes, underspecified requests, private reference requirements, and executable acceptance tests. A stateful user simulator separates task knowledge, disclosure rules, and persona while holding requirements fixed. Search, Experience, and Credence facets require direct clarification, behavioral evidence, and technical explanations, respectively (Nelson, 1970; Darby & Karni, 1973).

Our contributions are threefold. We formalize **interactive planning ability and its evaluation**, introducing PlanScore to measure discovery timing, resolution delay, and completion, complemented by report fidelity. We construct the **COSWE dataset and interaction protocol** from real repository changes, combining underspecified requests, private reference requirements, executable acceptance tests, and a stateful user simulator. Our **empirical analysis of planning performance and report quality** compares agents' discovery and resolution performance and shows that omissions exceed contradictions in final reports.

2 Related Work

Autonomous task completion. HumanEval, MBPP, EvalPlus, and LiveCodeBench evaluate code against executable tests, with increasing attention to test strength and temporal coverage (Chen et al., 2021; Austin et al., 2021; Liu et al., 2023; Jain et al., 2024). SWE-bench and Terminal-Bench broaden the unit of work to repository issues and command-line tasks (Jimenez et al., 2023; Merrill et al., 2026). InterCode evaluates interaction with execution environments, and SWE-agent studies interfaces for repository inspection and modification (Yang et al., 2023; 2024). These settings can require extensive tool use. COSWE adds user-owned decisions: execution feedback reveals what code does, whereas clarification establishes which behavior the user intends.

Long-horizon development. ReAct and Reflexion combine reasoning with actions or feedback-based revision (Yao et al., 2023; Shinn et al., 2023); DevEval, Commit0, and SWE-EVO study development activities, library implementation, and software evolution (Li et al., 2024; Zhao et al., 2024; Le et al., 2026). SWE-CI includes planning in an Architect–Programmer maintenance loop and evaluates evolving test outcomes (Chen et al., 2026). COSWE instead fixes a reference target while the agent acquires the user-owned decisions needed to formulate it.

User-interactive evaluation. Clarification research studies the value of questions (Rao & Daumé, 2018), while τ-bench and τ^2-bench evaluate collaborative tool–agent–user interaction (Yao et al., 2024; Barres et al., 2025). ICAE-Bench evaluates hidden-requirement coverage, repository correctness, and design quality; SWE-Interact evaluates both saved plans and implementation checkpoints against sub-goal rubrics (Peng et al., 2026; Raghavendra et al., 2026). These are direct precedents for interactive planning evaluation. Edwards & Schuster (2026) study uncertainty-aware clarification on underspecified software issues, analyzing query behavior alongside task resolve rates. COSWE makes the interval between discovering a particular decision and satisfying its acceptance conditions a scoring unit, alongside final-report alignment and executable outcomes. This complements progress and cost diagnostics in broader agent evaluation (Liu et al., 2024; Ma et al., 2024; Kapoor et al., 2024).

3 The COSWE Benchmark

3.1 Task and Planning Objective

A task is $b = (X, R, r, K, \mathcal{T})$, where X is the base repository and environment, R the complete reference specification, r the underspecified request, K the private knowledge base, and $\mathcal{T}$ the hidden acceptance and regression tests. The reference fixes a target solution $\pi^\star$ through intended technical decisions and their supporting requirements. Agents may inspect the repository to acquire implementation facts; the withheld facets instead concern user-owned choices such as scope, compatibility, boundary behavior, and acceptable trade-offs. Planning consists of identifying these missing decisions, satisfying the user's clarification requirements, and integrating the decisions into a coherent technical plan.

A scenario adds $\omega = (p, \delta, \beta)$: persona, disclosure rules, and interaction budgets. Separating these components permits communication behavior to vary without changing the reference target. A configuration that changes an accepted requirement is a task variant and requires corresponding reference and test updates. Initially, the coding agent receives X and r, and the user-facing language model receives r. The simulator backend holds K and controls subsequent disclosures. Neither participant receives the complete reference, reference patch, or hidden tests in the ordinary condition. Episodes interleave user messages, repository inspection, editing, and development testing, and are expected to produce a trace, a final `report.md`, and a patch. Available artifacts support evaluation of planning progress, report quality, and executable correctness.

3.2 Data Production Pipeline

The pipeline aligns each incomplete request, private knowledge base, and acceptance suite with one reference specification. Language models and author review assist authoring. Following the

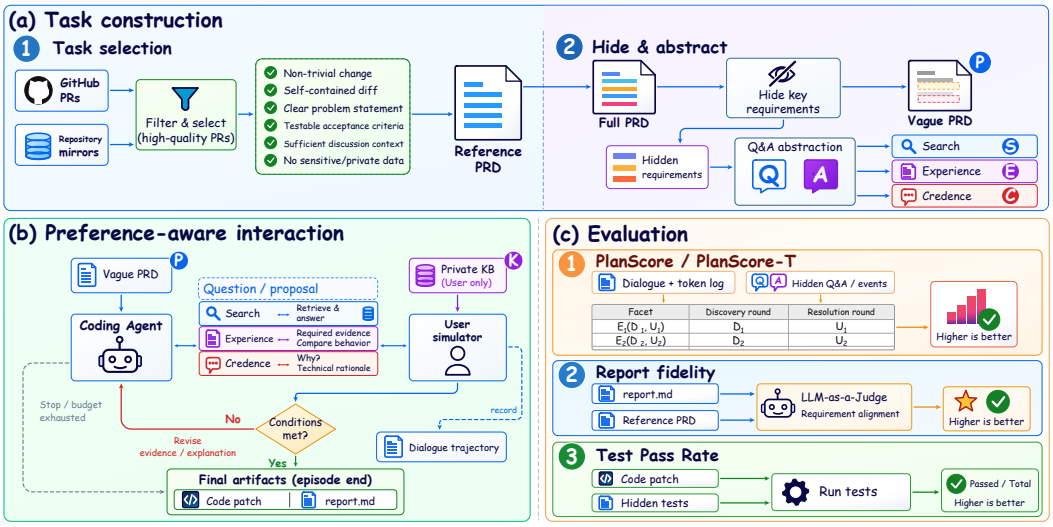

Figure 2: **The COSWE construction, interaction, and evaluation pipeline.** A reference specification links incomplete requests, private knowledge, and acceptance tests. The private KB feeds the simulator backend; the coding agent receives controlled replies. Accepted decisions and budget-based stopping are separate paths, and logging covers the interaction. Traces support PlanScore and PlanScore-T, reports support requirement-alignment fidelity, and hidden acceptance and regression tests assess patches.

emphasis on dataset documentation in Gebru et al. (2021), we record task provenance and distinguish required checks from completed validation. Appendix A gives the full protocol.

Figure 2 connects this authoring contract to controlled disclosure and evaluation. Semantic diffs and facet-to-test records make the alignment between disclosed answers and executable requirements reviewable.

1. Select and reconstruct repository changes. Merged pull requests provide patches, discussions, revisions, and timestamps. Author review selects consequential behavioral changes and excludes mechanical edits. The base revision is reconstructed with a repository-specific environment and test entry point. Reference and hidden-test patches are packaged separately and checked for applicability; authoring Git history is excluded from the exposed workspace.

2. Establish the reference contract. Source discussions, implementation changes, and tests are reconciled into `full_spec.md`. Atomic facets record intended decisions, accepted outcomes, provenance, and acceptance checks. The reference patch witnesses the authored contract; evaluation concerns accepted behavior rather than patch similarity. Our validation requires tests to accept the reference and reject the unchanged base for the requested functionality. This focus on observable capabilities is related to behavioral testing in CheckList (Ribeiro et al., 2020).

3. Create controlled underspecification. Authors create `vague_spec.md` by retaining goals and public context while removing decision values. A semantic diff links each omission to an answer-free question scope, reference answer, and provenance. These form K, with stable facet identifiers, S/E/C types, disclosure conditions, and verifier metadata, checked against the reference and acceptance tests.

4. Validate and package. Structural checks validate schemas and states; content review checks goal coherence, reference consistency, and answer disclosure. Executable checks validate the reference and test sensitivity. Optional controls and completed validation stages are recorded separately. Task packages retain the specifications, knowledge base, patches, environment, provenance, and

quality records; runtime views expose only authorized artifacts. Candidates and validated tasks remain distinct.

3.3 KNOWLEDGE-BASED USER INTERACTION

The S/E/C types define the interaction required to settle a facet, rather than three disjoint task subsets or a universal difficulty ordering. **Search (S)** entries permit a direct answer after a relevant question. **Experience (E)** entries require behavioral evidence or technical comparisons matching predefined cases and accepted outcomes. **Credence (C)** entries require explanations covering specified causal effects, risks, or trade-offs. The terms adapt distinctions about when users can assess qualities (Nelson, 1970; Darby & Karni, 1973); accepting an explanation is an operational criterion, not proof that credence uncertainty has disappeared.

A **Router–Gate–Voice** architecture separates interpretation, disclosure, and response generation. The Router maps atomic questions and submitted material to answer-free facet scopes and accessible knowledge. The Gate applies versioned rules: relevant S queries permit answers, E submissions are checked against required evidence coverage, and C explanations against predefined nodes. Unmet conditions yield requests for missing material; satisfied conditions unlock the reference content. The Voice expresses approved information according to the configured persona while preserving decisive content. S entries begin known to the simulator, though not necessarily disclosed; E/C entries begin latent. States, prior disclosures, evidence, and budgets persist across turns.

The gate evaluates submitted evidence, not whether a claimed experiment was actually run. Some C gates use an LLM judge with a regular-expression fallback. LLM-based judgments have documented biases and limitations (Zheng et al., 2023); our verifier versions are therefore part of the protocol and require validation. Still-latent E/C content fields are excluded from Router and Voice inputs, but clues in other visible material may still reveal a target decision. Routing, disclosure, and response fidelity therefore remain audit targets. A facet's disclosure and its correct implementation are distinct events.

Illustrative interaction. Figure 1 illustrates this interaction: the agent compares implementation options and clarifies restoration after the user selects a plan. Report fidelity evaluates whether the final report preserves reference requirements; hidden tests assess implementation.

Personas change communication without changing the reference answer. Persona, disclosure rules, and budgets are logged separately. Their configurability does not establish robustness across combinations; the supplementary user-mechanism pilot changes several components jointly.

4 EVALUATING INTERACTIVE PLANNING

4.1 DISCOVERY, RESOLUTION, AND PLANSCORE

A clarification round is one coding-agent message to the user; its reply shares the index. Multiple questions in one message count once, and repository tool actions do not advance the counter. For facet i, $D_i \geq 1$ denotes first effective discovery and $U_i \geq D_i$ resolution, when the decision becomes explicit and consistent with R. With valid logs, the discovery, resolution, and PlanScore are

$$d_i = \begin{cases} 100/D_i, & \text{discovered,} \\ 0, & \text{otherwise,} \end{cases} \quad r_i = \begin{cases} 100/(1+U_i-D_i), & \text{resolved,} \\ 0, & \text{otherwise,} \end{cases} \quad P_i = \sqrt{d_i r_i}. \tag{1}$$

Thus a resolved facet scores $100/\sqrt{D_i(1+U_i-D_i)}$; a discovered but unresolved facet retains discovery credit but receives zero PlanScore. This rewards early discovery and short subsequent delays, with completion required for speed credit. The geometric mean is computed within each facet, never from aggregated discovery and resolution means. We give eligible facets equal weight within each run, average repeated runs within a task, and then weight tasks equally. Unresolved facets remain in the denominator. Discovery and resolution diagnostics use the same aggregation. Clarification completion is the percentage of eligible facets resolved, with the same run–task averaging; it measures coverage rather than timing.

The present implementation evaluates eligible E/C facets using trigger and unlock events as proxies for discovery and resolution. S facets are outside this historical scoring set. Events indicate protocol

progress rather than independently verified understanding; missing or invalid logs remain unscored rather than being converted to failures. PlanScore measures interaction rounds and completion, not pure speed independent of coverage, wall-clock time, or implementation success.

Discovery in round 2 and resolution in round 4 yields $P_i \approx 40.82$; immediate resolution in round 2 yields 70.71, and nonresolution yields zero. Repository-tool actions do not advance rounds, motivating token-aware reporting alongside interaction speed.

4.2 Token Adjustment: PlanScore-T

PlanScore-T applies a relative token penalty to the same recorded planning outcomes. Let T_{mqij} be cumulative coding-agent token usage through resolution for model m, task q, facet i, and repetition j. Let $\mathcal{S}_{mqi}$ contain successful resolutions, J_{mq} count scored repetitions, and $\mathcal{M}_{\text{ref}}$ be the frozen reference pool. Define

$$\bar{T}_{mqi} = \frac{\sum_{j \in \mathcal{S}_{mqi}} T_{mqij}}{|\mathcal{S}_{mqi}|}, \quad B_{qi} = \min_{m \in \mathcal{M}_{\text{ref}} : |\mathcal{S}_{mqi}| > 0} \bar{T}_{mqi}, \quad G_{mqi} = \left[\min\left(1, \frac{B_{qi}}{\bar{T}_{mqi}}\right) \right]^{1/4}. \tag{2}$$

For evaluated tasks $\mathcal{Q}$ and N_q eligible facets per task,

$$P_m^T = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{N_q} \sum_{i=1}^{N_q} G_{mqi} \left(\frac{1}{J_{mq}} \sum_{j=1}^{J_{mq}} P_{mqij} \right). \tag{3}$$

The reference is specific to a task and facet. Unresolved repetitions contribute zero to the score mean but are excluded from the successful-token mean. Groups without any successful resolution contribute zero without evaluating the ratio. Facets unresolved throughout the frozen reference pool retain undefined references; unresolved groups remain in the task denominator. Missing token records are missing data, not zero usage. For matching scored outcomes, $0 \le P_m^T \le P_m \le 100$.

Token accounting includes deduplicated logical coding-agent input and output, cached input, and reported reasoning usage; it excludes the simulated user and judge. It measures cumulative usage through resolution, not marginal facet cost, total expenditure across failed attempts, elapsed time, or monetary cost. Reference values and providers are frozen by version, and the reference and evaluation pools overlap. Recalibration requires consistent rescoring. Section 6 separates the resulting score and rank changes from the primary planning comparison; Appendix C details the calibration rules.

4.3 Plan Quality and Executable Outcomes

Report fidelity evaluates the design expressed in a saved final report against requirements derived from the frozen complete reference. The semantic protocol `design-alignment-v2-full.1` uses GLM 5.3 to label each reference atom as aligned, contradicted, unmentioned, unresolved, or requiring review, with supporting reasons and report evidence. For report ρ, let $\mathcal{G}_\rho$ be its active requirement groups, A_g the frozen atoms in group g, and $y_{\rho a}$ the atom label. The fidelity score is

$$F(\rho) = \frac{100}{|\mathcal{G}_\rho|} \sum_{g \in \mathcal{G}_\rho} \frac{1}{|A_g|} \sum_{a \in A_g} \mathbf{1}[y_{\rho a} = \text{aligned}]. \tag{4}$$

Replacing aligned with contradicted, unmentioned, or unresolved gives the corresponding error component. Reports with any review-pending atom, missing reports, and unavailable references receive no score; they are never imputed as zero. We average scored repetitions within each task and then weight tasks equally, reporting scoring availability alongside fidelity. It evaluates the final expressed design, not a plan observed before implementation or agreement with every dialogue utterance. Appendix C gives label handling and evaluator provenance.

Test pass rate (TPR) is $100\, n_{\text{passed}}/n_{\text{tests}}$ for valid runs with nonempty designated test suites. The evaluator restores its hidden acceptance and regression assets and tests the final patch in an offline task container. TPR is a per-run test-case fraction, aggregated by averaging repetitions within tasks and then weighting tasks equally; complete task success requires all designated checks to pass. Code-induced failures are retained, while missing outcomes and infrastructure failures are recorded

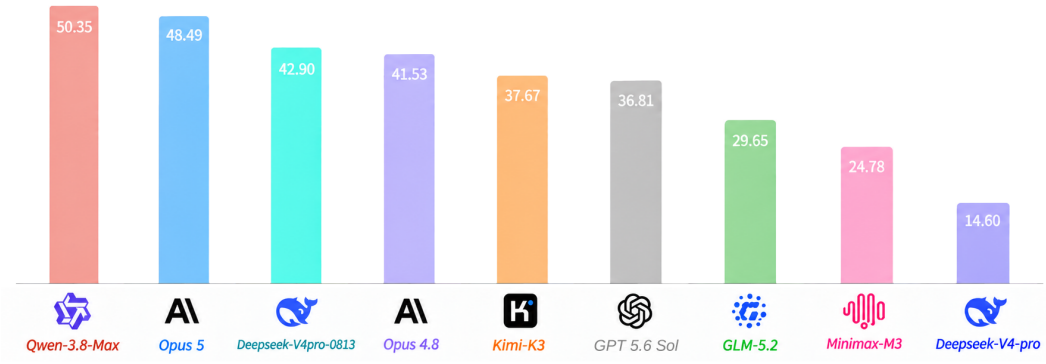

Figure 3: **PlanScore on the shared planning cohort.** Models are ordered by their task-macro PlanScore; labels give scores on the 0–100 scale. The resource-adjusted comparison is reported separately in Section 6.

separately. Passing supports the encoded contract, not correctness on all possible inputs (Liu et al., 2023). Associated traces, reports, patches, and evaluation logs connect clarification decisions to documentation and delivered behavior.

5 Experiments

We examine three questions: how efficiently agents resolve intended decisions, how discovery and resolution relate to planning scores and functional correctness, and how faithfully final reports preserve the reference design. Nine coding-model configurations are compared using archived evaluations and subsequent statistical reanalyses. The planning comparison holds task identities fixed; report fidelity uses a separate shared subset with scorable reports. We average repetitions within tasks and then weight tasks equally. Appendix B specifies evaluation-set selection, aggregation, and calibration provenance. Section 6 separately analyzes the frozen token adjustment. Uncertainty is interpreted at the sampling unit of each design (Dror et al., 2018).

5.1 Primary Planning Performance: PlanScore

Figure 3 evaluates the discovery and resolution of intended decisions before any token adjustment. Qwen 3.8 Max leads at 50.35, followed by Opus 5 at 48.49 and DeepSeek V4 Pro 0813 at 42.90. These scores combine early discovery, short resolution delays, and completion; they do not isolate pure interaction speed among completed facets.

Table 1A reports TPR for every model on the shared planning tasks. Opus 5 leads at 66.7%, while GPT 5.6 Sol is second at 64.8% despite ranking sixth in PlanScore. Appendix B.2 adds clarification completion. Unequal repetitions and unavailable uncertainty estimates preclude significance claims about adjacent scores.

5.2 Discovery, Resolution, and Test Pass Rate

Table 1 separates actual test pass rates (A) from correlations with TPR (B–C). Pearson r measures linear association; Spearman ρ measures rank association. In the available-task summaries, resolution relates more strongly to TPR ($r = 0.794$, $\rho = 0.767$) than discovery ($r = 0.365$, $\rho = 0.283$). Both scores combine coverage with timing, so this pattern does not isolate speed from completion.

Discovery and resolution also correlate with PlanScore at $r = 0.900$ and 0.962. These associations partly follow from $P_i = \sqrt{d_i r_i}$ and are not independent validation. PlanScore and PlanScore-T correlate with TPR at $r = 0.659$ and 0.679 on available tasks, versus 0.857 and 0.887 on shared tasks. Task composition and outcome eligibility differ between these analyses; their contrast cannot be attributed solely to fixing task identities.

Table 1: **Implementation outcomes and planning–test associations.** A reports test pass rates on the shared planning tasks. B and C report correlations with TPR across model means using each model's available tasks and the shared tasks, respectively. TPR values are percentages; r and ρ are correlation coefficients, computed from rounded means. The two correlation cohorts have different task and outcome eligibility.

A. Implementation results
Shared planning tasks

Model	TPR (%)
Qwen 3.8 Max	60.8
Opus 5	**66.7**
DeepSeek V4 Pro 0813	59.1
Opus 4.8	58.2
Kimi K3	52.5
GPT 5.6 Sol	64.8
GLM 5.2	52.8
MiniMax M3	48.7
DeepSeek V4 Pro	44.5

B. Available-task correlations

Metric	Pearson r	Spearman ρ
Discovery	0.365	0.283
Resolution	0.794	0.767
PlanScore	0.659	0.650
PlanScore-T	0.679	0.683

C. Shared-task correlations

Metric	Pearson r	Spearman ρ
PlanScore	0.857	0.767
PlanScore-T	0.887	0.817

Table 2: **Final-report requirement fidelity and error types.** Rates are percentages of reference requirements within each group, averaged equally over groups, scored reports within each task, and tasks. The four labels sum to 100% up to rounding. Missing or review-pending outcomes remain N/A.

Model	Fidelity (aligned) ↑	Contradicted ↓	Omitted ↓	Unresolved ↓
Qwen 3.8 Max	78.11	4.53	16.52	0.83
Opus 5	**83.01**	4.34	11.93	0.72
DeepSeek V4 Pro 0813	70.85	3.39	24.17	1.59
Opus 4.8	72.83	3.59	21.56	2.02
Kimi K3	80.53	3.63	15.10	0.74
GPT 5.6 Sol	70.59	1.11	25.47	2.83
GLM 5.2	69.69	5.63	21.04	3.64
MiniMax M3	60.20	11.17	27.14	1.49
DeepSeek V4 Pro	52.49	7.21	37.07	3.23

Within-model associations also vary (Appendix B.3). Opus 5's discovery, resolution, and PlanScore-T correlations with TPR are 0.482, 0.560, and 0.511; GPT's are 0.013, -0.076, and -0.112. Positive between-model associations therefore do not imply uniformly positive within-model associations. Full model-mean matrices remain in the data; missing paired records limit adjusted reanalysis. These descriptive associations do not establish a causal benefit from higher planning scores.

5.3 Fidelity of the Final Report

Table 2 assesses preservation of the reference design in final reports. Fidelity is the aligned-requirement rate; Contradicted denotes conflicts with the reference, Omitted maps to unmentioned requirements, and Unresolved denotes requirements discussed without enough detail to settle key decisions or conditions. These four label rates partition scored requirements. Pending reviews and missing artifacts are N/A, not report errors.

Opus 5 leads fidelity at 83.01%, followed by Kimi K3 at 80.53% and Qwen at 78.11%. Opus 5 receives scores for 64.86% of archived runs, versus 89.19% for Opus 4.8 and GPT, so fidelity is conditional on scoring availability. Omissions exceed contradictions for every model: GPT records 25.47% versus 1.11%, and DeepSeek V4 Pro records 37.07% versus 7.21%. This identifies incomplete requirement capture as a recurring failure. The report and planning cohorts are separate; their means do not establish trajectory-level fidelity–PlanScore relationships. Appendix C.3 details judging, missingness, and sensitivity; independent human agreement remains unestablished.

Table 3: **Effect of token adjustment on planning scores and ranks.** Ranks change from the PlanScore order (before) to the PlanScore-T order (after) on the same shared planning tasks.

Model	PlanScore	PlanScore-T	Rank before → after
Qwen 3.8 Max	**50.35**	30.50	$1 \rightarrow 5$
Opus 5	48.49	**36.34**	$2 \rightarrow 1$
DeepSeek V4 Pro 0813	42.90	28.51	$3 \rightarrow 6$
Opus 4.8	41.53	33.54	$4 \rightarrow 3$
Kimi K3	37.67	31.37	$5 \rightarrow 4$
GPT 5.6 Sol	36.81	33.77	$6 \rightarrow 2$
GLM 5.2	29.65	20.31	$7 \rightarrow 7$
MiniMax M3	24.78	19.07	$8 \rightarrow 8$
DeepSeek V4 Pro	14.60	11.37	$9 \rightarrow 9$

6 Token-Aware Planning

A short dialogue can conceal substantial repository-tool and reasoning work. We therefore treat PlanScore-T as a complementary resource view of the same interaction outcomes, keeping PlanScore as the primary planning comparison.

Table 3 compares scores and ranks before and after token adjustment. Opus 5 moves from second to first, Qwen moves from first to fifth, and GPT moves from sixth to second. Token adjustment can improve a model's rank when its score receives a smaller relative reduction than competitors' scores. Missing facet-level token ledgers prevent reconstructing whole-model token expenditure from these aggregates.

Two archived facet examples explain the rule. For xarray E1, Opus 5 uses a successful-resolution mean of 5,514,400 tokens against a reference of 4,333,000: $G = (4{,}333{,}000/5{,}514{,}400)^{1/4} = 0.94150$, taking a group PlanScore of 100 to 94.15. For Django E1, DeepSeek V4 Pro 0813 uses 951,300 tokens against 245,300: $G = 0.71260$ and PlanScore-T is 71.26. Thus a smaller absolute token count can receive a stronger penalty when it is further above its own facet reference. These examples illustrate normalization, not a decomposition of whole-model rank changes.

Reference support and penalty strength affect these comparisons. Some references rely on one successful repetition; alternative reference rules and exponents change intermediate ranks and can change the leader. Appendix D reports this sensitivity. Scores should therefore be interpreted with their frozen calibration rather than as invariant model properties.

7 Discussion and Conclusion

COSWE evaluates how agents discover user-owned decisions, satisfy acceptance conditions, and preserve the resulting designs. Qwen leads PlanScore, whereas Opus 5 leads PlanScore-T. Report errors are dominated by omissions. Different rankings and aggregation-dependent planning–test associations motivate evaluating interaction progress, expressed design, and implementation separately.

The protocol fixes acceptable designs, while real projects allow alternatives and changing goals. Gate acceptance does not establish understanding or verified execution; leakage and test-contract mismatches require auditing. Report fidelity depends on GLM 5.3 judgments and scoring availability without independent human agreement. Unequal repetitions, incomplete paired records, sparse token references, and joint changes in the user-mechanism pilot limit significance and causal conclusions.

AI Use Statement

Generative AI was used to assist benchmark authoring and to polish the manuscript. Language models also form part of the experimental framework for user simulation and semantic report-fidelity judging, as described in Sections 3.2–3.3 and Appendix C.3. The limitations of simulator acceptance and model-based judgments are discussed in Section 7.

Reproducibility Statement

Sections 3–6 and Appendices A–D specify protocols, metrics, and diagnostics. Accompanying data and scripts reproduce figures, model-mean correlations, and archived fidelity aggregation. Missing paired records limit trajectory-level reanalysis; independent semantic-label validation remains unavailable.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models, 2021. URL https://arxiv.org/abs/2108.07732.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment, 2025. URL https://arxiv.org/abs/2506.07982.

Jialong Chen, Xander Xu, Hu Wei, Chuan Chen, and Bing Zhao. SWE-CI: Evaluating Agent Capabilities in Maintaining Codebases via Continuous Integration, 2026. URL https://arxiv.org/abs/2603.03823v4.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code, 2021. URL https://arxiv.org/abs/2107.03374.

Herbert H. Clark and Susan E. Brennan. Grounding in Communication. In Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley (eds.), *Perspectives on Socially Shared Cognition*, pp. 127–149. American Psychological Association, 1991. doi: 10.1037/10096-006. URL https://psycnet.apa.org/record/1991-98452-006.

Michael R. Darby and Edi Karni. Free Competition and the Optimal Amount of Fraud. *The Journal of Law and Economics*, 16(1):67–88, 1973. doi: 10.1086/466756. URL https://doi.org/10.1086/466756.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1383–1392. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1128. URL https://aclanthology.org/P18-1128/.

Nicholas Edwards and Sebastian Schuster. Ask or Assume? Uncertainty-Aware Clarification-Seeking in Coding Agents, 2026. URL https://arxiv.org/abs/2603.26233v3. Version 3, September 7, 2026.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé, III, and Kate Crawford. Datasheets for Datasets. *Communications of the ACM*, 64(12):86–92, 2021. doi: 10.1145/3458723. URL https://arxiv.org/abs/1803.09010v8.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code, 2024. URL https://arxiv.org/abs/2403.07974v2.

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?, 2023. URL https://arxiv.org/abs/2310.06770.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI Agents That Matter, 2024. URL https://arxiv.org/abs/2407.01502.

Tue Le, Minh V. T. Thai, Dung Nguyen Manh, Huy Phan Nhat, and Nghi D. Q. Bui. Swe-evo: Benchmarking coding agents in long-horizon software evolution scenarios, 2026. URL https://arxiv.org/abs/2512.18470.

Bowen Li, Wenhan Wu, Ziwei Tang, Lin Shi, John Yang, Jinyang Li, Shunyu Yao, Chen Qian, Binyuan Hui, Qicheng Zhang, Zhiyin Yu, He Du, Ping Yang, Dahua Lin, Chao Peng, and Kai Chen. Prompting Large Language Models to Tackle the Full Software Development Lifecycle: A Case Study, 2024. URL https://arxiv.org/abs/2403.08604v3.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation, 2023. URL https://arxiv.org/abs/2305.01210v3.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.03688.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 74325–74362, 2024. doi: 10.52202/079017-2365. URL https://arxiv.org/abs/2401.13178v2.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces, 2026. URL https://arxiv.org/abs/2601.11868v1.

Phillip Nelson. Information and Consumer Behavior. *Journal of Political Economy*, 78(2):311–329, 1970. doi: 10.1086/259630. URL https://doi.org/10.1086/259630.

Zhongyuan Peng, Dan Huang, Chuyu Zhang, Caijun Xu, Changyi Xiao, Shibo Hong, David Lo, Lin Qiu, Xuezhi Cao, Jiyuan He, and Yixin Cao. ICAE-Bench: Evaluating Coding Agents as Interactive Project Builders, 2026. URL https://arxiv.org/abs/2607.21217v1.

Mohit Raghavendra, Anisha Gunjal, Aakash Sabharwal, and Yunzhong He. SWE-Interact: Reimagining SWE Benchmarks as User-Driven Long-Horizon Coding Sessions, 2026. URL https://arxiv.org/abs/2606.30573v1.

Sudha Rao and Hal Daumé, III. Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information. In *Proceedings of the 56th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2737–2746. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1255. URL https://aclanthology.org/P18-1255/.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.442. URL https://arxiv.org/abs/2005.04118.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366v4.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 23826–23854, 2023. doi: 10.52202/075280-1035. URL https://arxiv.org/abs/2306.14898.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering, 2024. URL https://arxiv.org/abs/2405.15793v3.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629v3.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains, 2024. URL https://arxiv.org/abs/2406.12045.

Wenting Zhao, Nan Jiang, Celine Lee, Justin T. Chiu, Claire Cardie, Matthias Gallé, and Alexander M. Rush. Commit0: Library Generation from Scratch, 2024. URL https://arxiv.org/abs/2412.01769.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023. doi: 10.52202/075280-2020. URL https://arxiv.org/abs/2306.05685v4.

Didar Zowghi and Chad Coulin. Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In Aybüke Aurum and Claes Wohlin (eds.), *Engineering and Managing Software Requirements*, pp. 19–46. Springer, 2005. doi: 10.1007/3-540-28244-0_2. URL https://doi.org/10.1007/3-540-28244-0_2.

A Task Artifacts and Stateful Interaction

Each task binds a reconstructed repository change to a frozen reference specification, an underspecified request, a private knowledge base, and executable acceptance criteria. This appendix specifies the authoring and access contract. Per-task records distinguish required checks from completed validation; describing a check does not imply that every archived task completed it. The reference fixes intended behavior, while interaction controls access to the decisions needed to formulate that behavior.

A.1 Source Reconstruction and Task Artifacts

Merged pull requests provide implementation changes, issue and review discussions, tests, and source timestamps. Author review selects consequential behavioral changes and excludes mechanical edits. The retained provenance identifies the repository, base and reference revisions, source discussions, and patches. Authors reconstruct the base revision with a repository-specific environment and test entry point, separate implementation changes from hidden evaluation tests, and check patch applicability. The exposed workspace excludes the local Git history used for authoring.

Authors reconcile these sources into `full_spec.md`, an authored reference rather than an assumption that the original issue was complete. Its decisions and supporting requirements define $\pi^\star$. The reference patch witnesses this contract; agents are evaluated on accepted behavior, not patch similarity. Authors then create `vague_spec.md` by retaining the goal and public context while removing selected decision values. A requirement-level redaction record links each omission to an answer-free question scope, a reference answer, and source provenance. This record connects the incomplete request to the private knowledge base.

Table 4: Task artifacts and their ordinary runtime visibility. Approved disclosures reach the coding agent through simulator replies; the private knowledge base is not an agent input.

Artifact	Purpose	Runtime access
Base repository and environment	Starting code, public context, and development tools	Coding agent
Underspecified request	Task goal with selected decisions withheld	Coding agent and user-facing model
Complete reference and reference patch	Frozen decisions and executable witness	Authoring and evaluation
Private knowledge base	Facet answers, disclosure conditions, and verification metadata	Simulator backend
Hidden acceptance and regression tests	Evaluate the final patch against the contract	Evaluator
Provenance and validation records	Reconstruct sources, checks, and artifact versions	Authoring and evaluation

A.2 Acceptance Checks and Validation Records

Executable validation requires the reference implementation to pass the designated acceptance and regression checks. The unchanged base is a negative control for the requested functionality. A missing new API may cause an expected import or collection failure; the execution context distinguishes this outcome from an infrastructure failure. Together, these checks assess reference executability and test sensitivity, without proving that all intended behavior is exhaustively tested.

Structural checks verify identifiers, required fields, states, and disclosure rules. Content review checks goal coherence, consistency between reference answers and tests, and accidental answer disclosure. Answer-string searches assist redaction review but cannot establish semantic isolation. Optional full-specification and no-interaction controls probe implementation from the complete reference and behavior under withheld information. These are authoring diagnostics, not additional results in the principal model comparison. Unperformed checks are recorded separately from failed checks; failure without interaction does not establish that interaction is logically necessary.

A.3 Knowledge States and Simulator Components

Each knowledge-base entry records a stable facet identifier, S/E/C type, answer-free question scope, knowledge state, reference content, disclosure condition, and verification metadata. Search entries begin `known` to the backend but need not have been disclosed to the coding agent. Experience and Credence entries begin `latent`. Experience entries specify behavioral cases, comparison formats, and accepted outcomes; Credence entries specify explanation nodes concerning causal effects, risks, or trade-offs. All types may occur within one task and do not impose a universal difficulty ordering.

The Router maps atomic questions and submitted material to facet scopes, currently accessible knowledge, and public evidence formats. Still-latent answer fields are excluded from its input. The Gate applies the versioned acceptance rules: relevant Search queries permit direct answers; Experience submissions must cover the required behavioral cases; Credence explanations must cover the specified nodes. Some archived Credence gates use an LLM judgment with a regular-expression fallback. The applicable verifier and fallback versions are part of the protocol. Acceptance checks submitted material; it does not independently verify that a claimed experiment was executed or that an explanation is causally correct.

The Voice expresses gate-approved information using the configured persona without changing decisive content. Still-latent answer fields are also excluded from Voice inputs, although clues elsewhere can defeat field-level isolation. Routing, leakage, and response fidelity therefore require auditing. Questions, submissions, disclosures, state transitions, and budgets persist in the interaction record. Unresolved facets may remain when an episode ends. Persona and disclosure settings govern communication; changing an accepted requirement instead creates a task variant requiring corresponding reference and test updates.

B Evaluation Protocol and Supplementary Results

B.1 Evaluation Sets and Aggregation

The planning and report evaluations use distinct archived task sets. The primary planning comparison retains the original shared task identities after model filtering, whereas all-available analyses use each model's eligible tasks. Report fidelity selects a separate common set with scored reports from every evaluated model. These sets are not interchangeable, and differences in eligibility can affect comparisons. The principal model comparisons retain nine coding configurations. The historical token reference pool remains frozen and retains ten coding configurations, including GLM 5.3; its separate report-judge role is disclosed in Appendix C.3.

Except where explicitly stated otherwise, a model's repeated runs are averaged within each task, and task means are then weighted equally. Report fidelity averages only scored repetitions; planning metrics retain unresolved facets as zeros when the underlying logs are valid. TPR averages per-run test-case pass fractions only for valid outcomes with nonempty designated test suites, following Section 4. Code-induced failures remain valid; missing outcomes and infrastructure failures are recorded separately and are not imputed as zero TPR. These different missingness rules reflect different observed quantities and must not be silently interchanged.

The planning definitions and relative-token results come from reports dated September 18 and September 21, respectively. The September 23 and September 24 association reports reanalyze the historical pool rather than introduce new coding-agent runs. Report fidelity uses the separate September 15 `design-alignment-v2-full.1` snapshot with references frozen before judging. The source manifest records report filenames, internal dates, collection dates, hashes, and their result mappings; collection dates do not imply new evaluations.

B.2 Completion and Executable Outcomes

Table 5 supplements the PlanScore and PlanScore-T comparison in Table 3 with clarification completion and implementation outcomes. Completion and TPR use the shared planning cohort but assess clarification and implementation, respectively.

Table 5: Shared planning-cohort outcomes. Clarification completion and TPR are task-macro percentages: completion measures resolved eligible E/C facets, while TPR averages per-run test-case pass fractions.

Model	Completion (%)	TPR (%)
Qwen 3.8 Max	72.4	60.8
Opus 5	**80.6**	**66.7**
DeepSeek V4 Pro 0813	64.0	59.1
Opus 4.8	60.9	58.2
Kimi K3	58.8	52.5
GPT 5.6 Sol	62.1	64.8
GLM 5.2	44.5	52.8
MiniMax M3	38.8	48.7
DeepSeek V4 Pro	25.0	44.5

B.3 Historical Within-Model Associations

Table 6: Archived within-model Pearson correlations with TPR. Each observation is a task mean after averaging matched valid trajectories within that model. The available task sets differ across models. These coefficients are retained from the source report, rather than re-estimated from unavailable paired observations.

Model	Discovery	Resolution	PlanScore-T
Qwen 3.8 Max	−0.190	−0.155	−0.080
Opus 5	0.482	0.560	0.511
DeepSeek V4 Pro 0813	0.201	0.342	0.124
Opus 4.8	0.267	0.225	0.305
Kimi K3	0.204	0.017	0.220
GPT 5.6 Sol	0.013	−0.076	−0.112
GLM 5.2	0.171	0.333	0.298
MiniMax M3	0.441	0.278	0.295
DeepSeek V4 Pro	0.241	−0.041	0.015

Table 6 shows that associations vary with the model and its task set. Opus 5 has positive coefficients for all three measures, while Qwen's are negative and GPT's resolution and relative-score coefficients are slightly negative. Discovery and resolution scores combine coverage with timing, so these correlations do not isolate speed conditional on completion. Between-model correlations can also reflect general capability differences and task composition; neither aggregation supports a causal effect of clarification on correctness.

The package contains full Pearson and Spearman matrices of rounded model means. Within-model coefficients remain archived estimates: missing paired records prevent task-demeaned or completion-adjusted re-estimation and trajectory-level confidence intervals (Dror et al., 2018).

C Scoring, Missingness, and Evaluation Validity

Section 4 defines the process, report, and implementation measures. The following details specify scoring eligibility, aggregation, and the frozen token calibration. The historical cohorts are not fully paired; joint analyses require matching the corresponding task and run records.

C.1 Planning Events and Aggregation

The archived planning estimates cover eligible E/C facets. Trigger and unlock events proxy discovery and resolution; they indicate protocol progress rather than independently verified understanding. A coding-agent message to the user advances one clarification round, regardless of how many questions it contains. Repository-tool actions do not advance this counter.

Equation 1 is applied within each facet before aggregation. Eligible facets receive equal weight within a run, scored repetitions are averaged within tasks, and tasks receive equal weight. Here J_{mq} counts repetitions with valid, scored planning logs. Undiscovered or unresolved facets in valid logs retain their prescribed zeros; missing or invalid logs remain unscored. Discovery and resolution diagnostics use the same weighting. These scores combine timing with coverage, so neither component measures speed conditional on completion alone.

The timing incentives in Section 4 are conditional: earlier discovery increases PlanScore at a fixed resolution delay, and a shorter delay increases it at a fixed discovery round. Holding the absolute resolution round fixed is a different comparison. For example, with $U_i = 8$, discovery at $D_i = 1$ or $D_i = 8$ gives $100/\sqrt{8} \approx 35.36$, whereas $D_i = 4$ gives $100/\sqrt{20} \approx 22.36$. Thus PlanScore is not monotone in discovery time when resolution time is fixed: later discovery also shortens the measured resolution delay.

C.2 Frozen Token Calibration

Equations 2–3 apply one coefficient to each model–task–facet group. Token usage is cumulative through resolution, with successful repetitions defining the token mean; failed attempts do not enter that mean. The frozen reference is the lowest successful model mean for the same task and facet. Its historical pool contains ten coding configurations, including GLM 5.3. The principal evaluation retains nine configurations. Removing a model from evaluation does not recalibrate the historical pool; GLM 5.3's separate report-judge role is documented below.

A group with no successful resolution contributes zero without evaluating a token ratio. In the historical calibration, facets unresolved by every model have undefined references and contribute zero because they have no successful resolutions. A future first resolution requires an explicit calibration rule and version before assignment of a relative score. New models scored against an existing reference retain the coefficient cap at one.

The source report describes recovering historical token accounting. The released aggregates permit summary verification, but their underlying per-call ledgers are unavailable for independent checks of deduplication and resolution cutoffs. Required records must be recovered before reporting matched PlanScore and PlanScore-T. Missing usage cannot be filled with zero or silently removed. If recovery is impossible, the treatment and resulting scoring population require an explicit versioned rule. The two reported scores must use matching scored outcomes and weights.

Write $G_{mqi} = [\min(1, B_{qi}/\bar{T}_{mqi})]^{\lambda}$. The default is $\lambda = 0.25$; the archived sensitivity analysis also reports 0.10 and 0.50. For positive P_m, with $w_{qi} = 1/(|\mathcal{Q}|N_q)$ and $\bar{P}_{mqi}$ averaging all scored repetitions including unresolved zeros,

$$\frac{P_m^T}{P_m} = \frac{\sum_{q,i} w_{qi} \bar{P}_{mqi} G_{mqi}}{\sum_{q,i} w_{qi} \bar{P}_{mqi}}.$$

Retention is therefore a PlanScore-weighted mean of coefficients, not a percentage of tokens saved. It cannot recover total expenditure or an average token multiple. Alternative reference rules and parameter sensitivity are described in Appendix D.

C.3 Semantic Report-Fidelity Protocol

Report fidelity evaluates the retrospective `report.md` against frozen reference requirements using `design-alignment-v2-full.1`. It does not observe a plan before implementation or establish grounding in every dialogue utterance. Evaluation-set selection and source provenance are described in Appendix B and the data manifest.

Reference construction and semantic labels. Necessary requirement atoms are extracted from the full PRD, checked against source text, deduplicated, and frozen before report judging. Background information, test names, and dataset-production details are not reporting obligations. GLM 5.3 reads each complete report and produces a label, reason, report line spans, and quotations for each active reference atom. The five labels are `aligned`, `contradicted`, `unmentioned`, `unresolved`, and `needs_review`. GLM 5.3 serves as the judge and is excluded from the nine

evaluated coding-model configurations. The source describes checks of document hashes, identifiers, quoted line spans, and response structure before deterministic aggregation.

Weighting and unavailable outcomes. Equation 4 weights atoms equally within each requirement group and groups equally within a report. Contradiction, unmentioned-requirement, and unresolved-requirement rates replace the aligned indicator with the corresponding label, retaining these same weights. Unresolved requirements may be discussed but leave key decisions or conditions insufficiently specified. If any atom is marked `needs_review`, the report's final fidelity is N/A pending review, even though provisional component counts remain available. Missing reports and missing references are likewise N/A, never zero fidelity. An unresolved requirement within an otherwise scored report contributes no aligned coverage; this differs from an unresolved judging decision, which makes the entire report unscored.

For model m, task t, and scored reports $\mathcal{R}_{mt}^{\text{scored}}$, the model score is

$$\bar{F}_m = \frac{1}{|\mathcal{Q}_F|} \sum_{t \in \mathcal{Q}_F} \frac{1}{|\mathcal{R}_{mt}^{\text{scored}}|} \sum_{\rho \in \mathcal{R}_{mt}^{\text{scored}}} F(\rho). \tag{5}$$

The error components use the same report, repetition, and task weights. Scoring availability is the number of scored runs divided by all archived runs on the selected task set. It is distinct from report-presence rate and from the fraction of tasks covered.

Common-cohort selection and unavailable reports. The main comparison sets $\mathcal{Q}_F$ to the intersection of tasks with at least one scored report from every retained coding model. Every selected task contributes one task mean for every model, while the number of scored repetitions may differ. The common selection therefore fixes task identities but remains conditioned on report availability and completed judgments.

Statuses are mutually exclusive: a record lacking both report and reference receives `reference_missing`, so report presence must be assessed independently from scoring status. Legacy manual and automatic records are excluded.

All-available sensitivity. Table 7 replaces the common task set with each model's own available scored task set. Qwen reaches 80.97% fidelity and Opus 5 reaches 80.19%, reversing their observed order in the common-cohort comparison. Because the task sets and retained repetitions change, this difference reflects sensitivity to the analyzed sample; it is not a temporal improvement or an unconditional model ranking. The report cohort is separate from the PlanScore/TPR cohort, and their model means are not used to infer fidelity–PlanScore or fidelity–TPR correlations without a task- and run-level join.

Table 7: All-available report-fidelity sensitivity. Each model uses its own available scored task set, so differences reflect both model behavior and task coverage. Task identities therefore differ from the common-cohort comparison.

Model	Fidelity (%)
Qwen 3.8 Max	80.97
Opus 5	80.19
DeepSeek V4 Pro 0813	74.77
Opus 4.8	74.02
Kimi K3	78.68
GPT 5.6 Sol	72.33
GLM 5.2	71.16
MiniMax M3	59.60
DeepSeek V4 Pro	49.38

Reproduction and judgment validity. The released numerical evidence reproduces group rates, scored-run handling, and task-macro model summaries from archived atom labels and requirement-group records. All scored and review-pending records identify GLM 5.3 as judge. This agreement establishes arithmetic consistency, not semantic label correctness. Independent human-consensus

labels and a full second-judge evaluation are unavailable, and the archive lacks the exact judge prompt and generation settings.

The package contains opaque task/run aliases, status flags, per-group label counts, snapshot hashes, and a reproduction script. It omits report and reference text, quotations, and original identifiers. Independent semantic assessment requires those underlying documents and a fully specified judging protocol.

Execution and artifact availability. Hidden acceptance and regression tests are restored before evaluating patches. Code-induced failures remain valid; missing outcomes and infrastructure failures are recorded separately. Task/run identifiers link available traces, reports, patches, and test logs, but incomplete archives limit joint analyses.

D Calibration Sensitivity and Exploratory User Comparison

D.1 Alternative Reference Rules

The default calibration retains the historical reference pool rather than re-estimating it from the evaluation cohort. The supported-reference alternative requires at least two successful resolutions and a success rate of at least 50%; facets without an eligible alternative retain their available default reference. Sparse successful references limit the stability of fine-grained relative scores, and fallback means that the supported-reference condition is not uniformly based on stronger evidence. Section 6 illustrates the task- and facet-specific normalization.

D.2 Sensitivity of PlanScore-T Rankings

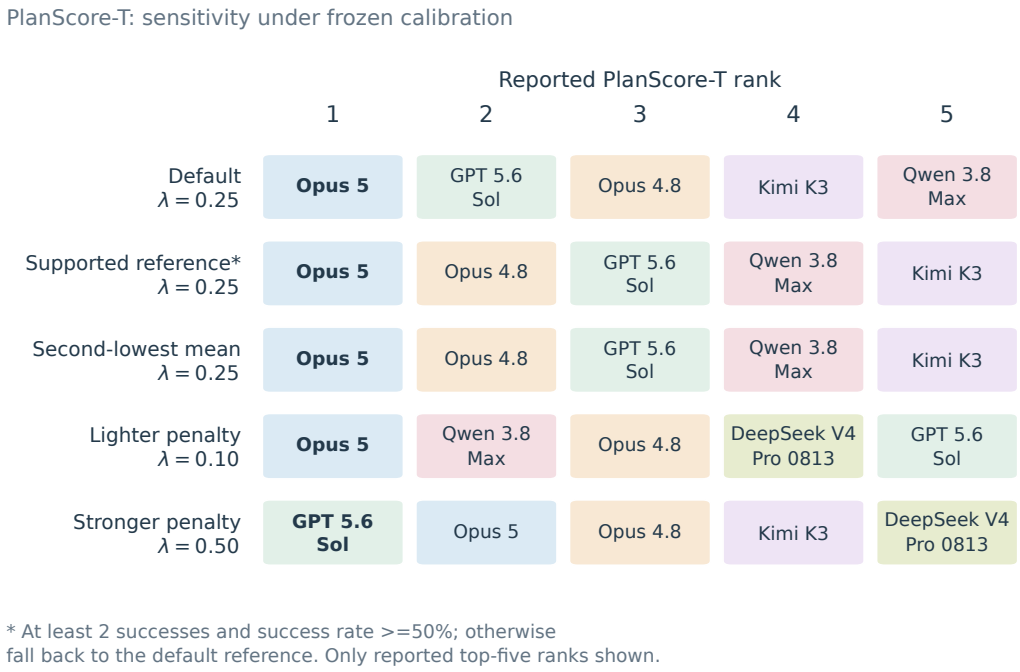

Figure 4: Archived top-five orders under alternative token-reference rules and penalty exponents. The supported-reference condition retains the default where no eligible alternative exists, so it is not a uniformly high-support ranking. Only reported positions are available.

Opus 5 leads under the default and alternative reference rules and the lighter penalty; GPT leads under the stronger penalty. These archived top-five orders show configuration sensitivity, not repeated-run uncertainty. Missing full rankings and per-facet fallback records prevent independent reconstruction of every alternative.

D.3 Exploratory Comparison of User Mechanisms

Table 8: Exploratory user-mechanism comparison. Mean TPR is computed over comparisons with matching initial coding-agent requests, weighting model–task combinations equally. The conditions differ in multiple user-mechanism components.

User mechanism	Mean TPR (%)
KB-gated user	60.45
Full-specification user	66.65

The pilot contrasts a knowledge-base-gated user with a user given the complete reference; the coding agent initially receives an underspecified request in both arms. The full-specification user is an experimental control that changes the ordinary user-access condition in Section 3.1; it does not give the coding agent the complete reference at initialization. Table 8 summarizes comparisons whose saved initial coding-agent requests match. The aggregate weights model–task combinations equally; when a task appears with different models, these combinations receive separate weights.

The observed aggregate difference is 6.20 percentage points. Knowledge representation, routing, response generation, and disclosure requirements change jointly; matching saved requests does not establish identical historical environments. This exploratory comparison therefore does not isolate a causal effect of gating or information visibility.