

Cross-Provider Review as a Runtime Contract for Coding Agents: A Controlled Pilot and Fault-Injection Study

Bowen Xu^{*†}

Department of Biology, Stanford University, Stanford, CA, USA

bxu666@stanford.edu

Boyu Chen^{*}

Institute of Health Informatics, University College London, London, UK

**Equal contribution. [†]Corresponding author.*

Abstract

Coding agents increasingly share a workstation while drawing on separate providers and subscription allowances. A second agent can inspect a completed answer, but the call spends another pool and may provide no substantive finding. We describe an advisory cross-provider review contract: distinct resource pools, bounded execution, restricted reviewer capabilities, complete input delivery, usable semantic output, explicit failure states and durable per-attempt evidence. In a controlled, agent-authored pilot of 20 paired development turns, eight had a material reviewer finding (95% exact interval 19.1–63.9%). A boundary-condition scan across both reviewer backends reproduced a previously discovered false success on partial input: four truncation levels passed historically and failed after repair. The scan also found and repaired cancellation during process reaping. In real CLI probes, Claude had no writing tools; Codex attempted writes in five of five read-only trials, each write tool failed, and no disposable repository changed. These tests cover specified paths and versions, not field reliability. A preregistered shadow study of metadata-only review allocation accrued 25 formal observations before an exact-runtime regression found a third defect: a reviewer exiting nonzero with a well-formed verdict was counted as complete. Exit status was not recorded per attempt, so exposure cannot be resolved retrospectively. The 25 formal and two pending records remain an audit cohort; the measurement-valid cohort restarted at zero and collection has begun. No gate result is reported.

1 Introduction

A coding agent can read a repository, run commands, change files and tell a developer that a task is done. A second model from another provider may notice a consequential mistake in that answer. In a workstation with several subscriptions, however, the second call draws on a distinct, exhaustible allowance. A fixed policy that reviews every completed turn pays this cost even for status updates, direct questions or refusals. A policy that never reviews loses the chance to catch important errors. A practical system needs both a reliable way to obtain an independent review and evidence about where the review can be spared.

The first step is easily understated. Calling another model is not a trustworthy review unless the system can show that the reviewer received the complete material, produced a usable response, and was actually independent in provider and resource pool. Process exit code alone does not establish those properties. The review also needs an explicit relationship to the primary answer: in our setting it is advisory, returned after the answer has been delivered, and cannot silently rewrite the repository. The resulting outcome is a reviewer finding, not observed improvement in final software quality.

Testing that contract exposed defects in its own implementation. A synthetic scan found that cancellation during process reaping could still be counted as a completed review. Later, after preregistered collection had begun, an exact-runtime regression found that a nonzero reviewer exit accompanied by a well-formed verdict

was also counted as complete. A zero exit status does not establish a review, and a well-formed verdict does not redeem a failed process: both are necessary and neither is sufficient. Because exit status had not been recorded per attempt, the affected observations could not be scoped after the fact.

We study this setting in OpenTerminalAgent (OTA),¹ a terminal-agent runtime developed by Bowen Xu that runs Codex, Claude, Kimi and other coding CLIs behind one interface while each draws on its own provider allowance. The paper makes three bounded contributions:

1. A cross-provider advisory review contract for completed coding-agent turns that makes pool disjointness, execution bounds, input delivery, non-mutation, typed failure and durable per-attempt evidence part of the measurable interface.
2. A completed controlled pilot of 20 paired coding-agent turns, with two provider directions and turn-level blinded source checking. It demonstrates review can surface material issues in this workload and measures how often a uniform policy yielded no material finding.
3. A deterministic fault-injection case study of this reviewer lifecycle, with repair-before/after comparisons for three implementation-specific false-success paths, one discovered during a live preregistered collection, plus a preregistered shadow allocation protocol. Neither is a claim that selective gating already works.

This is a systems-and-evaluation contribution. We do not claim a new routing algorithm, a calibrated risk model, or broad generality from 20 turns.

2 Related Work and Boundary of the Contribution

Selective verification, cost-sensitive cascades and model routing are established ideas (Aggarwal et al., 2024; Chen et al., 2023; Ong et al., 2024). SEVRA selects verification from solver-visible states (Dip et al., 2026); solve-versus-verify work allocates compute (Singhi et al., 2025). Our V3 gate is zero-shot: prospective labels did not exist at freeze, and pilot training could learn its selected task mix. This constraint has no demonstrated advantage. A separate subscription pool reviews a delivered answer without replacing it. We measure missed material findings against avoided calls, not final-answer accuracy under one token budget.

SWE-agent and OpenHands provide repository/tool environments (Yang et al., 2024; Wang et al., 2024), Agentless a simpler validation path (Xia et al., 2024), and SWE-bench issue-level tasks (Jimenez et al., 2023). Review assistance raises trust concerns in a field study (Aðalsteinsson et al., 2025). On 116 tasks, Xiang et al. (2026) find directional asymmetry in Claude/Codex review, including regressions after review changes an answer; Song (2026) finds partly different planted errors with cross-model and fresh same-model review. Shaw (2026) finds plausible but incorrect security results from missing payloads and audit evidence; harness frameworks make observability and validation runtime duties (Zhong & Zhu, 2026; Ahn & Kim, 2026). Cross-model review also has an official tool: OpenAI (2026) provides a plugin that lets Claude Code call Codex to review code or delegate tasks. Guo et al. (2026) find that independently checked evidence helps weaker patch reviewers and model size alone does not predict review quality; Lin & Zhang (2026) model AI screening before human approval as a capacity decision. Gao (2026) shows through fault injection across coding agents that recorded events need not faithfully represent skill-lifecycle boundaries. Our narrower question is whether each post-delivery advisory review is a valid observation; the three false-success paths reported here are specific to this implementation.

Adjudicators can err: studies find self-recognition, self-preference and position biases (Panickssery et al., 2024; Shi et al., 2024; Chen et al., 2024). Our Claude-family pilot judge sees Claude-family text in both directions: the Codex → Claude critique and the Claude → Codex answer. Identity blinding and source checks reduce cues but establish neither model-family independence nor human ground truth.

¹<https://openterminalagent.surge.sh/>

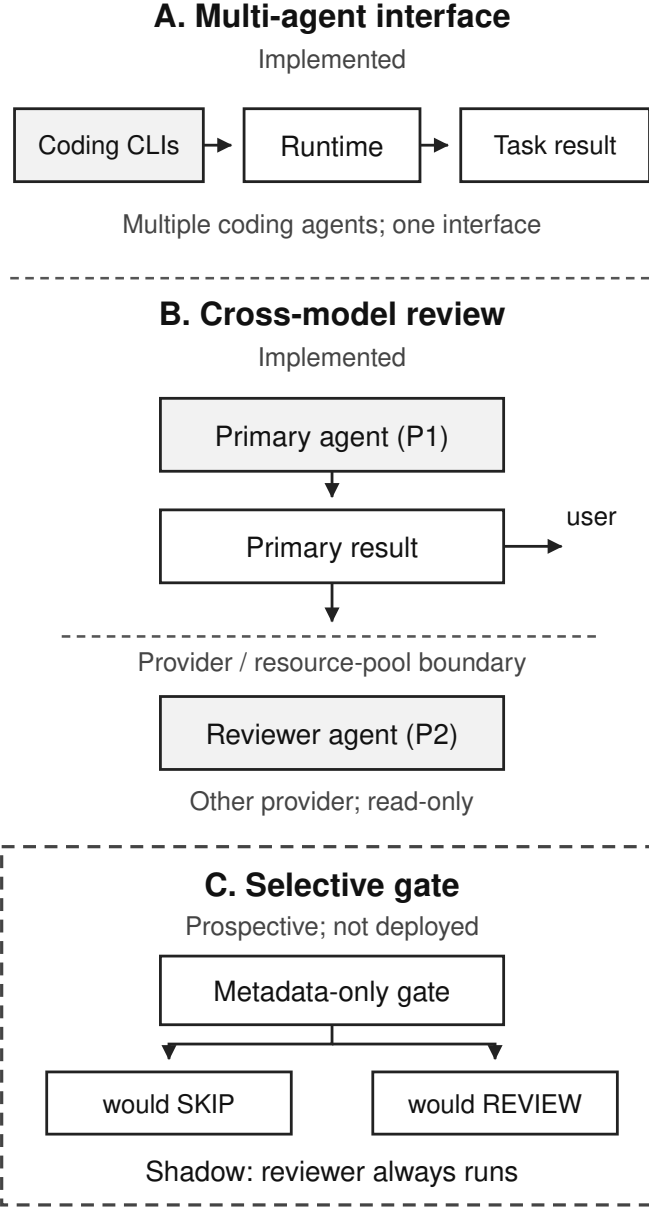


Figure 1: Implemented agent interface and cross-provider review versus the prospective metadata-only gate. In shadow evaluation, the reviewer still runs on every eligible turn. The gate corresponds to OTA’s Allocator role and the reviewer to its Reviewer role.

3 A Review Contract for Heterogeneous Coding Agents

The turn is one user instruction, one primary agent’s execution and its terminal answer. The reviewer receives the completed turn, operates in a separate process, and returns a critique. A review cannot retroactively change the answer. The runtime records a semantic outcome in addition to the process outcome; a successful process exit with no usable critique is not a completed review.

The contract has six observable properties (Table 1). A review counted as complete requires all applicable checks; merely obtaining process exit code zero is insufficient.

Table 1: Runtime contract, injected failure and measurement protected.

Property	Failure exercised	Protected measurement
Distinct resource pool	Same-pool configuration refused before spawn	Independent reviewer-call cost
Bounded lifetime	Child does not drain input; cancellation during review	Attempt denominator and timeout/cancellation attrition
Restricted capabilities	Write-enabled launch arguments rejected; real CLI asked to write in disposable repos	Observed working-tree integrity under tested backends
Complete input delivery	Plausible clean verdict after four truncation levels, from 1% delivered to one character missing	Valid review denominator
Semantic result and typed failure	Empty, malformed, exit-zero empty output, and nonzero exit with well-formed verdict	No failure miscounted as "no material finding"
Durable per-attempt evidence	Exit status not recorded when defect found after collection	Retrospective scoping of affected observations

This contract matters to the later gate study. A missing review cannot be treated as a turn with no material finding, because that would make the gate appear safer. A reviewer whose input was cut off cannot supply a valid label. A same-pool review would not measure the distinct-budget decision the study asks about. The contract turns such cases into explicit exclusions or failures rather than silent observations.

The reviewer always ran in the pilot. In the prospective gate study it will still run for every eligible turn. The gate operates in shadow mode: it produces a would-SKIP score and never suppresses a live call. Any unavailable, malformed or non-finite gate result yields no would-SKIP recommendation. We call this a fail-to-review decision for the *gate*; it is not a claim that a failed reviewer can be converted to approval.

OTA’s interface now exposes these functions as named roles: Primary Agent, Reviewer, an optional Allocator that gives metadata-only review advice, and an optional Judge that assesses a completed answer and its critique. Both optional roles are off by default, and Allocator advice never suppresses a review. These controls postdate the experiments below, whose role assignments were fixed by the experimental protocol.

4 Controlled Pilot: Method

We exercised the normal runtime path over 20 development tasks involving read-only analysis of a real codebase. The tasks covered architecture, bug localisation, code review, test gaps, configuration, subprocess life cycle, performance, security and related engineering questions. They were authored under controlled conditions by an agent, not sampled from ordinary users. There were 23 execution attempts; 20 produced paired complete primary answers and completed reviews. Only those 20 enter the materiality denominator. The two provider directions each contain ten paired turns; task order was not randomised between directions.

The materiality unit was a turn, not a count of comments. A turn was MATERIAL if accepting a reviewer finding would substantively improve correctness, safety, reproducibility, implementation or architecture validity, an important omitted constraint, a false technical claim or a meaningful failure-mode assessment. Mainly stylistic or optional suggestions were NON_MATERIAL. UNCERTAIN was available when evidence did not support a reliable decision. All 20 pilot items were labelled by one fresh-context Claude Opus 5 subagent, once per item. It received shuffled, opaque packets with task, answer and critique, without model identity, direction, cumulative counts or hypothesis; factual claims had to be checked against source. This is a model adjudication, not an independent human audit. In the Codex-primary/Claude-reviewer direction, the adjudicator shares the reviewer’s model family; in the reverse direction it shares the primary’s. Either overlap may affect the 2/10 versus 6/10 contrast. There is no inter-adjudicator agreement estimate. A keyword classifier recorded by the product was not used as ground truth.

Table 2: Controlled pilot by provider direction; medians in seconds.

Direction	Pairs	Mat.	Nonmat.	P med.	R med.
Codex → Claude	10	2	8	77.7	117.5
Claude → Codex	10	6	4	282.9	24.7

The reviewer prompt and decision rule were fixed throughout. We recorded primary and reviewer latencies, reviewer process outcomes, repository mutation and leaked-process checks. These engineering checks establish that the contract ran; they are not an estimate of user-perceived latency or of review’s effect on final code.

5 Controlled Pilot: Findings

Eight of the 20 paired turns had at least one material reviewer finding; 12 had none, and none was labelled UNCERTAIN. The exact two-sided 95% Clopper–Pearson interval for 8/20 is 19.1–63.9%. This is a wide interval on a selected controlled workload. It shows that review found substantive issues in this sample and that a uniform reviewer ran on twelve turns without a material finding. It does not show a 40% population rate or that twelve calls could safely have been skipped in advance.

The directions differed in task/answer mix, order and runtime behaviour. The count difference is not an interpretable ranking of provider or reviewer quality, especially with the family-overlapping adjudicator. We expose the asymmetry rather than infer a treatment effect.

If simple Claude-family self-preference dominated materiality labels, it would tend to favour the Claude critique in Codex → Claude (raising MATERIAL) and the Claude answer in Claude → Codex (lowering MATERIAL). The observed 2/10 and 6/10 run opposite to those directional predictions. This weakens that single-factor explanation of the contrast, but does not rule out judge bias, task-mix effects or other confounding.

The reviewer launched in all 23 attempts and completed on all 20 paired turns; across the attempts the recorded audit found no repository mutation, leaked process, timeout or cancellation. One observed primary allowance exhaustion left the other provider’s reviewer able to run; an attempt with no complete primary answer was correctly declined. This is a single operational example of pool separation, not a reliability estimate.

An audit during this work found a concrete input-integrity defect: partial delivery could previously be mistaken for a successful review if the reviewer replied. The contract now treats delivery completeness as a prerequisite to counting a review. This experience illustrates why review trustworthiness depends on the harness around the model as well as on the model’s judgement.

6 Fault Injection and CLI-Level Write Probe

We exercised the production reviewer lifecycle with both the Claude structured-result parser and Codex JSONL assistant-item parser in a detached checkout, using in-memory child processes and a synthetic 300,000-character input. The oracle counts only a nonempty semantic verdict as a completed review. Distinct boundary conditions, rather than repeated identical trials, probe four delivered-input lengths, three cancellation phases and three output times around a one-second deadline for each backend. No formal V3 data or provider call entered this synthetic scan.

The input comparison reproduces a specific historical false success across four truncation boundaries in each backend. The cancellation scan found a second defect: after output was read, cancellation during process reaping still allowed a clean completion. The runtime now polls the original deadline and cancellation state through reaping and before accepting the verdict. An exact-runtime regression later exposed a third path: a reviewer that exited nonzero after emitting a complete, well-formed verdict was counted as a completed review in every runtime revision used for collection. The post-repair candidate treats any nonzero exit as

Table 3: Boundary scan and real CLI probe. Each synthetic boundary is one deterministic case; CLI rows are five distinct disposable repositories per backend.

Condition	Observed classification or filesystem outcome
Clean-looking verdict after 1%, 50%, 99%, or all but one character delivered	Both backends: historical completed 4/4; current input_delivery_failed 4/4
Cancellation during input write, output read, or process reap	Both: pre-reap-repair accepted the reap case; current cancelled in all three phases
Nonzero exit with complete, well-formed verdict	Both: every collection-era revision completed; post-repair candidate process_crash ; all six exit/output scenarios classified as specified per backend (12/12)
Output at 0.8, 1.0, or 1.2 s with 1.0 s budget	Both: completed, timeout , timeout , respectively, in this timed scan
Malformed JSON; exit-zero empty result	Claude: malformed_output , empty_response ; Codex: empty_response for either; no clean verdict
Same-pool or write-enabled launch configuration	same-pool refusal or unsafe_argv , before launch
Valid full-input synthetic control	Completed
Real Claude Code 2.1.278, plan/no-tools configuration; explicit write request	Writing capability removed; 0/5 repositories changed
Real Codex CLI 0.155.1, read-only sandbox; explicit write request	5/5 attempted file changes and shell writes; all write tools failed; 0/5 repositories changed

an operational failure regardless of output. The exact-deadline case depends on scheduling; its observed class is reported, not treated as a statistical rate. The real CLI probes sent only synthetic instructions to temporary Git repositories, comparing file lists and a pre-existing file hash before and after. Claude’s no-tools configuration removes the means to issue a write call. Codex’s JSONL trace shows **file_change** and shell-write attempts in 5/5 trials, each ending in failed tool events under its read-only CLI mode. The event stream did not provide a permission-denial reason, so it does not establish which lower-level mechanism rejected the writes. Both findings are narrower than a universal non-mutation guarantee. A synthetic primary trace without a final answer also fails the readiness predicate, but full primary-cancellation integration is not established.

7 Prospective Shadow Study: Review Allocation

V3 pins Qwen3 models (0.6B primary; 1.7B and 4B controls) that see only task class, length/activity/time buckets and error/retry flags, excluding answers, findings, source text, paths and credentials. Frozen tokens and revisions define paired next-token would-SKIP scores, not calibrated probabilities. In OTA’s role terms, Qwen and Jev, an external commercial decision model queried by API as a secondary comparator affecting neither stopping nor execution (Almeida, 2026), are Allocators; the cross-provider CLI is the Reviewer, and blinded Kimi the Judge. Research adjudication is separate from the optional product Judge.

Every eligible turn receives review; nine frozen thresholds define offline would-SKIP sets. Scoring failures remain in the denominator and cannot propose a skip. Kimi source-checks sealed answers and findings, blind to scores, thresholds and cumulative results; its backend revision is undisclosed and qualified human audit unavailable.

The primary descriptive endpoint is MATERIAL among adjudicated 0.6B would-SKIP turns at $t=0.90$, with an exact interval. Fixed outputs include curves/failures by model/stratum, MATERIAL frequency and fraction skipped, and activity/rule baselines. The frozen hypergeometric random-skip benchmark expects $K \cdot M / N$ misses for K skips among N eligible turns with M MATERIAL labels. No outcome-selected threshold becomes primary.

Regardless of label, collection stops at 80 adjudicated reference would-SKIP turns or 400 eligible adjudicated measurement-valid turns, whichever comes first. Rare skips may leave an empty or imprecise safety endpoint at the cap, a reportable result. No V3 gate result is reported.

Complete-input repair and the three-state check predate V3. Exact-runtime audits confirmed reaping dead-lines but found Section 6’s nonzero-exit defect in every collection revision. Missing exit records prevent exposure scoping; misclassification is possible but not established. Under a disposition rule frozen before any label was inspected, the 25 formal and two pending records are kept unchanged as an audit cohort and excluded from efficacy analyses. Measurement-valid collection restarted at zero on a requalified runtime recording exit status and CLI-reported model per attempt. As of 2026-10-07, eight post-repair turns had been formalized, each with exit status and CLI-reported models recorded and matching the pinned values; no adjudication label had been inspected. Amendments frozen after stopping-counter visibility but before label inspection add matched coverage and a sparse-operating-point fallback; the original threshold remains primary.

8 Discussion and Limitations

The pilot shows that a reviewer from a separate pool ran on completed turns and surfaced material findings under a model-based rubric, and the synthetic scans show that the contract rejects the tested false-success paths; neither establishes general security or a causal benefit of review. Real CLIs left ten disposable repositories unchanged despite write requests. Safe recognition of the twelve non-material turns remains unknown, motivating shadow scoring and matched-count baselines. The third defect shows that checking without durable recording prevents later audit.

The read-only, agent-authored pilot has 20 pairs on one codebase in two provider directions. Wide intervals, nonrandom order, one judgement per item and partial adjudicator independence limit rate and provider comparisons. A natural-use stratum is preregistered but not currently collected; post-repair turns are agent-driven tasks drawn from one fixed product revision, and collection ends administratively if distinct tasks run out. V3’s gates share one model family; the adjudicator’s backend revision is undisclosed and lacks independent human audit. Proprietary conversations and reviewer instructions cannot be released unchanged. The gate has not yet been validated.

The contract has observable failure semantics, and the study design can register a negative result; independent human audits and calibrated decision models remain future work.

The authors used Codex for code, drafting and review, and Claude (Claude Code and claude.ai) for code review, drafting and review; they verified all analyses and claims.

9 Conclusion

In this setting, cross-provider review is measurable when the runtime checks complete input delivery, separate pools, bounded execution, restricted capabilities and explicit failures, and durably records that evidence for every attempt. The controlled pilot yielded material findings on eight of twenty turns. Boundary injection reproduced false successes on partial input and cancellation during reaping, and an exact-runtime regression found a third, on nonzero exit with a well-formed verdict, after a preregistered collection had begun. Repairs for all three paths pass synthetic regression, and post-repair collection has begun on the requalified runtime. Real read-only CLI probes observed no repository writes in five trials per backend. A preregistered shadow study will test whether a cheap metadata-only gate can allocate these second opinions better than matched random and transparent rules. No V3 effectiveness claim is made.

Author Contributions

Bowen Xu: Conceptualization, Methodology, Software, Investigation, Formal analysis, Visualization, Writing – original draft, Writing – review & editing, Project administration. Xu conceived and led the study.

Boyue Chen: Methodology, Supervision, Writing – review & editing.

Both authors contributed equally to this work. Bowen Xu is the corresponding author.

Competing Interests

Bowen Xu develops OpenTerminalAgent, the runtime studied in this paper, and may benefit from its adoption. Its source code, prompts and orchestration logic are proprietary and are not released; this paper describes only interfaces, the review contract and the evaluation protocol.

References

- Fannar Steinn Adalsteinsson, Björn Borgar Magnússon, Mislav Milicevic, Adam Nirving Davidsson, and Chih-Hong Cheng. Rethinking Code Review Workflows with LLM Assistance: An Empirical Study, 2025. URL <https://arxiv.org/abs/2505.16339>. arXiv:2505.16339.
- Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqi, and Mausam. AutoMix: Automatically Mixing Language Models, 2024. URL <https://arxiv.org/abs/2310.12963>. NeurIPS.
- Joongho Ahn and Moonsoo Kim. From Prompts to Contracts: Harness Engineering for Auditable Enterprise LLM Agents, 2026. URL <https://arxiv.org/abs/2607.08028>. arXiv:2607.08028.
- Diogo Almeida. Introducing System One Models & Jev, 2026. URL <https://typesafe.ai/blog/introducing-system-one-models-and-jev>. TypeSafe AI, Sep. 15.
- Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or LLMs as the Judge? A Study on Judgement Biases, 2024. URL <https://arxiv.org/abs/2402.10669>. arXiv:2402.10669.
- Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance, 2023. URL <https://arxiv.org/abs/2305.05176>. arXiv:2305.05176.
- Sajib Acharjee Dip, Dawei Zhou, and Liqing Zhang. Think Again or Think Longer? Selective Verification for Budget-Aware Reasoning, 2026. URL <https://arxiv.org/abs/2606.19808>. arXiv:2606.19808.
- Xueping Gao. Evidence-Calibrated Runtime Reconstruction for Agent Skills Across Heterogeneous Coding Agents, 2026. URL <https://arxiv.org/abs/2608.08793>. arXiv:2608.08793.
- Junyu Guo, Shangding Gu, Ming Jin, and Javad Lavaei. Groundability, Not Scale Alone: When Weak Reviewers Can Audit Strong Coding Agents, 2026. URL <https://arxiv.org/abs/2610.01023>. arXiv:2610.01023.
- 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>. arXiv:2310.06770.
- Ruihan Lin and Jiheng Zhang. When to Screen, When to Bypass: LLM-Judges in Resource-Scarce AI-Human Workflow, 2026. URL <https://arxiv.org/abs/2603.13870>. arXiv:2603.13870v2.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to Route LLMs with Preference Data, 2024. URL <https://arxiv.org/abs/2406.18665>. arXiv:2406.18665.
- OpenAI. Codex plugin for Claude Code, 2026. URL <https://github.com/openai/codex-plugin-cc>.
- Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM Evaluators Recognize and Favor Their Own Generations, 2024. URL <https://arxiv.org/abs/2404.13076>. arXiv:2404.13076.
- Animesh Shaw. Silent Failures in Agentic Security Evaluation: A Validated Harness for Tool-Call Mediation Under Indirect Prompt Injection, 2026. URL <https://arxiv.org/abs/2609.32691>. arXiv:2609.32691.

-
- Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the Judges: A Systematic Study of Position Bias in LLM-as-a-Judge, 2024. URL <https://arxiv.org/abs/2406.07791>. arXiv:2406.07791.
- Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When To Solve, When To Verify: Compute-Optimal Problem Solving and Generative Verification for LLM Reasoning, 2025. URL <https://arxiv.org/abs/2504.01005>. arXiv:2504.01005.
- Tae-Eun Song. When Does a Second Model Help? Cross-Model Review in LLM Verification, 2026. URL <https://arxiv.org/abs/2610.01471>. arXiv:2610.01471.
- Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An Open Platform for AI Software Developers as Generalist Agents, 2024. URL <https://arxiv.org/abs/2407.16741>. arXiv:2407.16741.
- Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based Software Engineering Agents, 2024. URL <https://arxiv.org/abs/2407.01489>. arXiv:2407.01489.
- Zuodong Xiang, Yike Zhang, YueMing Zhang, and Hailu Xu. Cross-Model LLM Code Review: Should you use Claude to review Codex or vice versa?, 2026. URL <https://arxiv.org/abs/2607.21656>. arXiv:2607.21656.
- 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/2405.15793>. arXiv:2405.15793.
- Hailin Zhong and Shengxin Zhu. AI Harness Engineering: A Runtime Substrate for Foundation-Model Software Agents, 2026. URL <https://arxiv.org/abs/2605.13357>. arXiv:2605.13357.

A Appendix: Post-repair requalification and operational record

This record covers qualification and collection status as of 2026-10-07 (UTC). It reports checks and counts only. No post-repair adjudication label or allocation result is reported.

A. Defects found during qualification and operation

1. An exact-runtime regression found that a nonzero reviewer exit could pass with a well-formed verdict; exit status was not durable per attempt, so the existing cohort was preserved for audit and new collection restarted with explicit exit evidence.
2. A model-provenance audit found that configured model choices did not prove which models the CLIs had used; CLI-reported model identities and their evidence hashes were added before the first post-repair turn.
3. A startup audit found error paths that could open a blocking dialog in unattended use; unattended error handling was changed and tested before collection began.
4. A project-directory preflight failed because import-time thread suppression also blocked subprocess output readers; the import allowlist was repaired before the first post-repair submission.
5. A prelaunch deadline check found that parsing could discard the UTC offset and shift the collection window; timezone-aware parsing and round-trip tests fixed it before collection began.
6. A selector-model check failed when a single-line JSON response was indexed as an array; whole-document parsing was repaired and replay-tested without changing the saved selection or existing turn records.
7. A valid `NO_DISTINCT_TASK` response reached an unhandled branch and paused collection; a disposition rule was frozen before labels were opened, and existing records were preserved.
8. A bootstrap self-test used the real data location and failed; it was moved to temporary synthetic data, and before/after file hashes confirmed no change during the checked runs, although creation-time whole-file hashes were unavailable for four early files.
9. A later preflight counted retained source rows of formalized turns as pending and stopped the supervisor; an operational patch was tested separately, with existing records preserved and collection left stopped.

B. Durable evidence for each attempt

Attempt receipts record exit status, the runner's termination reason, `STARTED` and terminal times, deadline and cancellation state, output validity, input-delivery completeness and eligibility. They also record each CLI's reported main model and all reported model identifiers, hashes of the supporting metadata fragments, and hashes of key runtime source files. A missing or mismatched required field makes the attempt ineligible. These checks establish recorded process and model provenance, not the correctness of the answer or critique.

C. Administrative counts and stopping

As of 2026-10-07, eight post-repair turns were formalized. All eight had recorded zero reviewer exit status, CLI-reported main models matching the pinned choices, strict-cohort receipts and matching source-file hashes. The selector had returned `NO_DISTINCT_TASK` seven times in total; the current consecutive count was zero. One other attempt was excluded because the Claude reviewer allowance was blocked. No further attempt attrition type was recorded in the two post-repair operating windows.

Only the controlled agent-driven stratum was active; the preregistered natural-use stratum was not enabled. The administrative rule ends the controlled stratum after three consecutive `NO_DISTINCT_TASK` returns if no task-source extension was frozen beforehand. With no natural-use collection, this ends the cohort at its

actual formal count. The checkpoint is descriptive, reports strata separately and uses matched coverage at 10%, 20% and 25% of that count. It computes no Phase-2 threshold below 50. Integrity checks and sealed label-blind scoring precede any label join. This administrative rule does not change the scientific stopping rule in Section 7.

D. Known limits

The frozen selection schema records how many duplicate ideas were rejected, not their contents or links to earlier selections. `SELECTION_BLOCKED` has no reason field. The Kimi CLI records the requested model but supplies no resolved backend model identifier. Candidate workspaces contained historical research artifacts; a path-level inventory found no adjudication label or outcome files, but this does not show that other artifacts were never accessed. An operator search exposed historical score text; the deviation record states that no label was opened and no exposed score informed a decision. This revision did not reread those scores.

Attempt and pending records lack direct candidate and task identifiers. The eight formal turns were linked through the live workspace/session association, then turn identifiers and sealed-source evidence. The candidate-to-turn mapping for the later pending captures remains unverified; these captures have not been formalized. Their chronological order was not accepted as proof of identity.