How a Maker-Checker, Test-Driven Process and a Single-Agent Ticket Run Differ: A Second Preregistered, Blinded Comparison on Seventeen Replayed Tourbot Tasks

Jacob Stephens

Educational Travel Adventures · stephens.page

Experiments 3–5 November 2026 · Public edition 17 November 2026 · ← Narrative essay (30-pair synthesis) · ← First study (10-issue adoption test)

In brief

Educational Travel Adventures (ETA) ran a second preregistered, paired, blinded comparison between its current way of running AI coding agents - thirdshift's Built-in Ticket run (the **Baseline**) - and a multi-agent, maker-checker test-driven workflow proposed for adoption (the **Candidate**). Where the first comparison (30 October – 1 November 2026; ten tasks) was an adoption decision with fixed win and cost thresholds, this second comparison is descriptive: it tests which parts of the Candidate workflow may be worth evaluating as modular changes to the Ticket run, using the Candidate's Codex-orchestrated configuration, its updated multi-agent TDD and cross-family review stages, seventeen newly selected replayed Tourbot tasks (plus three repeated tasks, for twenty pairs in all), a structured five-criterion rubric, a six-family panel of model-based graders plus two same-model diagnostic graders, blind human grading by the author across all delivered pairs, and independent blind sample reviews of two pairs (Pair 10 `T10` and Pair 12 `T12`) by a colleague.

Table A. Summary of the two arms across the 17 first-pair tasks and 3 repeats.

Measure	Baseline (`ticket-run`)	Candidate (Maker-Checker TDD)
Pull requests delivered (of 17 first pairs)	**17 of 17** (`16` green CI, `1` CI-watch failure on `T12`)	**11 of 17** (`10` green CI, `1` CI-watch failure on `T12`)
Non-delivery on 17 first pairs: `stopped-to-ask` / `failed`	**0** `stopped-to-ask` / **0** non-delivery	**4** `stopped-to-ask` (`T1`, `T8`, `T13`, `T16`) / **2** `failed` (`T3` timeout, `T9` workflow push)
Author's blind merge call on 11 delivered first pairs: `as_is` / `small_fixes` / `no`	**10 / 1 / 0** (`12 / 1 / 0` across all 13 graded pairs)	**2 / 6 / 3** (`2 / 7 / 4` across all 13 graded pairs)
Author's blind pair preference (11 delivered first pairs / all 17 first pairs)	**8 wins** (`3` ties) / **14 wins** (`3` ties)	**0 wins** (`3` ties) / **0 wins** (`3` ties)
Colleague blind sample review (2 pairs: `T10`, `T12`)	**`1 merge as is (T12)`, `1 small fixes (T10)`** (**1 win** on `T12`, `1` tie on `T10`; praised stronger input checks on `T10`)	**`0 merge as is`, `2 small fixes`** (**0 wins**, `1` tie on `T10`; praised separate test-file organization on `T10`)
7-voter combined panel (6 LLMs + author; 11 delivered / 17 first pairs)	**10 wins** / **16 wins**	**1 win** (`T10`) / **1 win** (`T10`)
6-LLM panel (11 delivered / 17 first pairs)	**5 wins** (`5` ties) / **11 wins** (`5` ties)	**1 win** (`T10`) / **1 win** (`T10`)
5-LLM OpenAI-excluded panel (`sonnet`, `gemini`, `grok`, `muse`, `kimi`; 11 delivered)	**8 wins** (`2` ties)	**1 win** (`T10`)
4-LLM family-excluded panel (`gemini`, `grok`, `muse`, `kimi`; 11 delivered)	**5 wins** (`6` ties)	**0 wins** (`6` ties)
Median API-priced cost per task (17 first pairs; paired median ratio **4.62×**)	**$1.27** (total `$22.50`; `$25.93` across 20 pairs)	**$5.46** (total `$177.07`; `$199.13` across 20 pairs)
Median adjusted time per task (17 first pairs; paired median ratio **3.28×**)	**18.9 min** (total `311.4 min` across 17 pairs)	**55.1 min** (total `1,686.7 min` across 17 pairs)

Measure	Baseline (`ticket-run`)	Candidate (Maker-Checker TDD)
Median test lines / production lines (11 delivered first pairs)	**163** test / **67** prod (**2.64×**)	**454** test / **56** prod (**10.10×**; paired test ratio **2.32×**)
Repeat self-consistency (3 repeated tasks: `T17` , `T8` , `T15`)	**2 of 3** same delivery outcome; **2 of 2** same human preference	**3 of 3** same delivery outcome; **1 of 2** same 6-LLM panel call

Key questions

1. **Why run a second comparison?** The first comparison rejected full adoption of the Candidate workflow (`2` wins in `10` , median `10.4×` cost, `6.4×` time; `1` of `8` Candidate pull requests mergeable as is against `7` of `8` Baseline), but left open four questions that this design addresses (Section 1, Section 2):

   (a) how the Candidate behaves when orchestrated by Codex `gpt-6.1-sol` instead of Claude Code (Section 2.3, Section 3.1, Section 4.4, Section 5),

   (b) how 64 subsequent commits to the Candidate's maker-checker TDD stage and updates to its cross-family review stage change its behaviour (Section 1.1, Section 2.3, Section 4.1, Section 4.4, Section 5.1),

   (c) whether the six themes observed in the first comparison hold as preregistered hypotheses on a larger, reproducibly selected task set (Section 1.2, Section 2.4, Section 4.1), and

   (d) how human and model-based graders compare when they score the exact same structured rubric fields (Section 2.6, Section 3.2, Section 4.2).

2. **What changed in the apparatus?** Both arms now use Codex `gpt-6.1-sol` (effort `high`) as their top-level orchestrator under `thirdshift 0.19.0` . Tasks are selected by a deterministic selection script (`17` tasks plus `3` seeded repeats = `20` pairs). Each task has a pre-refined, numbered acceptance checklist, and each grading packet adds base-test execution (`X_tests_on_base` , `Y_tests_on_base`) alongside cross-tests and Infection mutation testing. Six panel graders from six model families (`claude-sonnet-5-5` , `gpt-6-astra` , `gemini-4-argon-high` , `grok-4.7` , `muse-wonder-full-1.0` , `moonshotai/Kimi-K3.1-Pro[1m]`) and two diagnostic graders (`claude-opus-5-5` , `gpt-6.1-sol`) score each pair in both presentation orders on the same five-criterion rubric (Section 2.3–2.6).

3. **What happened on the six preregistered hypotheses (H1–H6)?** Four hypotheses (**H2**, **H3**, **H4**, **H6**) held strongly; two (**H1**, **H5**) did not show a Candidate advantage on this 17-task census (Section 4.1):

   (H1) Candidate cross-tests found **0** confirmed defects in Baseline pull requests across the 11 delivered pairs (while a Baseline cross-test caught 1 regression in the Candidate's `T17` pull request);

   (H2) the Candidate wrote a median **2.32×** as many test lines per task (`10.10` test lines per production line vs `2.64` for Baseline);

   (H3) the Candidate failed to deliver on **6 of 17** first-pair tasks (`4` `stopped-to-ask` , `1` 11.7-hour verification loop timeout on `T3` , `1` workflow-scope push failure on `T9`) whereas the Baseline delivered pull requests on all **17 of 17**;

   (H4) the Candidate performed **4.0×** as many file-read calls (`2,934` vs `729`), took a median **6.11×** longer to reach its first PHP edit (`11.4` vs `1.8` minutes), and spent **74.5%** of its primary-pair API-priced cost in orchestration, planning and RED test stages;

   (H5) neither arm showed a systematic fail-safe advantage on unexpected inputs; and

   (H6) the human maintainer preferred the smaller diff on **8 of 8** non-tied pairs (`0` for the larger diff), mainly penalizing excessive or implementation-coupled tests.

4. **Which parts of the Candidate workflow are worth evaluating in the Ticket run?** Switching the Candidate's orchestrator to Codex halved the cost-and-time gap relative to the first comparison (`4.62×` cost and `3.28×` time vs `10.4×` and `6.4×`), yet the Baseline was still preferred by the blind human reviewer on **8 of 11** delivered pairs (`3` ties, `0` Candidate wins) and by the 7-voter panel on **10 of 11** (`16 of 17` overall). Rather than adopting multi-session maker-checker loops or open-ended meta-review routing, the specific mechanisms worth testing as small, bounded additions to the Ticket run are (a)

an explicit base-test check (`tests_on_base`) verifying that new regression tests fail on the unpatched parent commit, (b) a single cross-family code review pass alongside Codex's review, and (c) hard circuit breakers and test-proportionality rules in review prompts (Section 5).

Abstract

Background. In our first preregistered comparison on ten replayed Tourbot issues (30 October – 1 November 2026), `thirdshift` 's single-agent Built-in Ticket run (the Baseline) outperformed a multi-agent, maker-checker test-driven workflow (the Candidate) on cost (median `$1.35` vs `$14.06`), time (`11` vs `62` minutes) and blind human merge-readiness (`7` of `8` vs `1` of `8` mergeable as is), while the Candidate's independently written tests exposed defects in two Baseline pull requests. In that first study, however, we ran the Candidate under a Claude Code orchestrator on an earlier revision of its process skills, selected tasks through an unrecorded intermediate shortlist, used two model graders that shared exact model IDs with the arms, and formulated our six explanatory themes after seeing the results.

Objective. To test six preregistered hypotheses (H1–H6) about how the Built-in Ticket run and the updated, Codex-orchestrated maker-checker TDD workflow differ in work, outcomes, failure modes and grader preferences across twenty replayed pairs (seventeen tasks plus three repeats), weighted towards blind human judgement, in order to identify which specific mechanisms of the Candidate are worth evaluating as additions to the Ticket run.

Methods. Preregistered, paired, blinded descriptive comparison. A deterministic selection script identified all 17 closed Tourbot tasks across both study windows (1 July – 3 November 2026) that had a single merged production pull request changing 2–8 files and +13 to +330 lines without migrations or staging-only work, were not in the first comparison, and carried an explicit acceptance-criteria section when labelled `ready-for-agent` . Three tasks (`T8` , `T15` , `T17`) were drawn with a recorded seed (`20261103`) for a second independent pair to measure run-to-run variation. Before the runs, the author refined each task's acceptance criteria into a frozen numbered checklist checkable inside a blind packet. Each arm attempted each task unattended in parallel on isolated 8-vCPU cloud droplets (`thirdshift 0.19.0`), with the Candidate running its updated maker-checker TDD and cross-family review stages under a Codex `gpt-6.1-sol` (`high`) orchestrator that delegated verification and review roles to Claude `claude-opus-5-5` (`medium`) and Codex `gpt-6.1-sol` sub-agents. For every completed pair, the grading harness formatted changed PHP files with `php-cs-fixer` , ran cross-tests (each pull request's tests on the other's code), base-tests (each pull request's tests on `base/`) and scoped Infection mutation testing inside an unprivileged grading account, and invoked eight model-based graders in both presentation orders (`[X, Y]` and `[Y, X]`): a six-family panel (`claude-sonnet-5-5` , `gpt-6-astra` , `gemini-4-argon-high` , `grok-4.7` , `muse-wonder-full-1.0` , `moonshotai/Kimi-K3.1-Pro[1m]` , none sharing a model ID with either arm) and two same-model diagnostic graders (`claude-opus-5-5` , `gpt-6.1-sol`), all at maximum CLI effort. The author graded every delivered pair blind before inspecting any model verdicts or arm mappings, and isolated blind portals were provided for colleagues, with one colleague recording independent blind unaided reviews on two pairs (Pair 10 `T10` and Pair 12 `T12`).

Results. Across the 17 first-pair tasks, the Baseline delivered a pull request on all 17 (`16` with green CI; median cost `$1.27` , median time `18.9` minutes, median `4` sessions), whereas the Candidate delivered a pull request on 11 (`10` with green CI), paused for human input on 4 (`stopped-to-ask` : `T1` , `T8` , `T13` , `T16`), timed out after `699.4` minutes (`64` sessions, `$64.22`) in unbounded meta-review loops on 1 (`T3`), and failed delivery after modifying a CI workflow file on 1 (`T9`). Across the 17 first pairs, the Candidate's median paired cost ratio was `4.62×` (`$5.46` vs `$1.27` ; total `$177.07` vs `$22.50`) and its median paired time ratio was `3.28×` (`55.1` vs `18.9` minutes). On the 11 delivered first-pair pull requests (`13` delivered pairs including repeats), the blind human reviewer rated the Baseline `merge as is` on `10 of 11` (`12 of 13` with repeats) and `merge after small fixes` on `1 of 11` , with `0` `don't merge` ; he rated the Candidate `merge as is` on `2 of 11` , `merge after small fixes` on `6 of 11` (`7 of 13`), and `don't merge` on `3 of 11` (`4 of 13`), preferring the Baseline on `8 of 11` first pairs (`10 of 13` with repeats; `3` ties, `0` Candidate wins). The 7-voter panel (`6 LLMs + author`) preferred the Baseline on `10 of 11` delivered first pairs (`1` Candidate win on `T10` , `0` ties; `16 of 17` first pairs including non-deliveries), while the 6-LLM panel preferred the Baseline on `5 of 11` (`1` Candidate win, `5` ties; `8 Baseline, 1 Candidate, 2 ties` when excluding OpenAI's `astra`). Across the colleague's two independent blind sample reviews (`1` Baseline win, `0` Candidate wins, `1` tie), the colleague rated both pull requests `merge after small fixes` (`tie`) on Pair 10 (`T10` - the lone

primary pair where the LLM panel and the author disagreed, crediting the Baseline's stronger input checks while preferring the Candidate's separation of slide-override tests into a dedicated test file) and rated the Baseline `merge as is` versus the Candidate `merge after small fixes` on Pair 12 (`T12` , matching the author's blind call where the 6-LLM panel tied). On H1–H6, Candidate cross-tests detected `0` confirmed defects in Baseline code (H1), while the Candidate wrote `10.10` test lines per production line vs `2.64` for the Baseline (H2), spent `74.5%` of its primary-pair API-priced cost outside implementation (`56.1%` across all 20 runs in the orchestrator alone; H4), and lost every non-tied human comparison to the smaller-diff Baseline pull request (H6).

Limitations. All seventeen tasks are small-to-medium changes (2–8 files, +13 to +330 lines in the shipped fix) to one legacy PHP codebase, starting from issues already shaped as `ready-for-agent` briefs rather than from unshaped ideas. The Candidate's Codex-to-Claude role recipes had to be reconstructed from its general role execution rules because a dedicated Codex-routing specification remained unimplemented at freeze time. Only one human grader (the author) scored all 13 delivered pairs, with a second blind human reviewer scoring 2 sample pairs (`T10` and `T12`).

Conclusion. Running the Candidate workflow under a Codex `gpt-6.1-sol` orchestrator with its updated 3 November rules narrowed the cost and time multiples from `10.4×` and `6.4×` (first comparison) to `4.62×` and `3.28×` , but did not close the delivery, merge-readiness or test-proportionality gap with `thirdshift` 's Built-in Ticket run (`17/17` vs `11/17` delivered; `10/11` vs `2/11` mergeable as is; `8–0–3` blind human preference). `thirdshift` should retain the single-agent Ticket run as its default implementation workflow and evaluate targeted, bounded mechanisms - specifically automated `tests_on_base` verification, a single cross-family review pass, and strict review circuit breakers - in place of multi-round maker-checker orchestration.

Keywords: coding agents, multi-agent systems, maker-checker, test-driven development, cross-family review, agent transcripts, LLM-as-a-judge, paired comparison, preregistration, replay evaluation, self-preference bias, run-to-run variance.

---

1 Introduction

1.1 Why a second comparison was needed

In our first comparison (Runs 30 October – 1 November 2026), we evaluated whether a multi-agent, maker-checker test-driven workflow [13] (combining a maker-checker TDD stage and a cross-family review stage) should replace `thirdshift` 's single-agent Built-in Ticket run on Tourbot, ETA's production PHP 8 / MySQL 8 operations platform. That first study was structured as a gatekeeping adoption test with preregistered thresholds: to replace the Ticket run, the Candidate had to win at least 8 of 10 pairs (or 13 of 18), keep median cost and time within 2× of the Baseline, and incur no more correctness failures. It was rejected on every threshold: it won 2 of 10 pairs, cost a median 10.4× as much (`$14.06` vs `$1.35` at API rates), took a median 6.4× as long (`62` vs `11` minutes), had 3 correctness or non-delivery failures to the Baseline's 2, and was judged mergeable "as is" by the blind human reviewer in only 1 of 8 delivered pull requests, against 7 of 8 for the Baseline.

Yet the most useful findings of the first comparison were not the pass/fail thresholds, but the exploratory observations from the twenty agent transcripts and sixteen pull requests:

1. **Independent tests caught real defects.** In two of the ten pairs from the first study, tests written from the issue before code was written exposed genuine behavioral defects in the Baseline's shipped pull requests.
2. **Verification loops multiplied test volume.** Across the eight graded pairs, the Candidate wrote a median 6.0× as many test lines as the Baseline (14.7 test lines per line of production code vs 3.7), often coupling tests to private helpers or exact SQL/HTML strings that the issue did not specify.
3. **Unanswered planning questions stopped unattended runs.** In two pairs, the Candidate stopped without opening a pull request because a planning or setup question went unanswered, whereas the Baseline made a sensible default assumption, documented it, and shipped.

4. **Fresh contexts re-read the codebase repeatedly.** The Candidate started a median 4.9× as many agent sessions and spent most of its time and tokens before making its first edit to production code (median 24 minutes to first code edit vs 1.3 minutes).
5. **Defensive branches and fail-safe handling differed.** The Candidate's code added more guards for unexpected inputs, though it still missed an environment-specific encoding constraint (`latin1` vs UTF-8) that no role thought to check.
6. **Human and model graders valued opposite things.** Model graders rewarded exhaustive test suites and defensive checks; the human maintainer penalized test bloat and unnecessary surface area in 7 of 8 Candidate pull requests.

Because those six findings were formulated *after* inspecting the first comparison's traces and scores, they were exploratory hypotheses rather than preregistered tests. Moreover, four methodological limitations in the first comparison complicated their interpretation:

- **Orchestrator harness and evolving skills.** At the 29 October freeze used in the first study, the maker-checker TDD specification supplied orchestration recipes only for Claude Code, so we ran the Candidate's orchestrator on Claude Code (`claude-opus-5-5`). Its author noted afterward that he normally runs the workflow under a Codex (`gpt-6.1-sol`) orchestrator, and between 29 October and 3 November he pushed 64 additional commits to the maker-checker TDD stage (adding roughly 1,100 lines, including model-checking/experimentation rules, bounded stopping rules, and pure-deletion/rename handling) and updated the cross-family review stage (adding round diagnoses and bounded meta-review routing) [3].
- **Task selection reproducibility.** The first comparison's written selection rule yielded 77 closed tasks when re-run on 3 November, whereas the original research step had worked from an unsaved intermediate shortlist of 24.
- **Same-model grader bias and coarse rubric output.** The first comparison used two model graders (`claude-opus-5-5` and `gpt-6.1-sol`) that were the exact models used inside the two arms, and each grader returned only `{winner, reason}` . When a model grader disagreed with the author, we could not tell from structured fields whether they disagreed on factual acceptance criteria, unexpected edge cases, codebase fit, test quality, or how those criteria should be weighed [6, 8].
- **Single-run noise.** Each arm attempted each task once, leaving unmeasured how much a process differs from itself on the same task versus how much it differs from the other process [9].

1.2 Objective and preregistered hypotheses

Rather than repeating an all-or-nothing adoption vote, we designed this second comparison to answer a constructive engineering question: **which parts of the Candidate workflow are worth evaluating as changes to `thirdshift` 's Built-in Ticket run?**

To answer that question rigorously, we preregistered the six explanatory themes from the first study as directional hypotheses (**H1–H6**), each with a pre-specified operational measurement across the 17 first-pair tasks:

- **H1 (Independent test fault detection).** *The Candidate's independently written tests expose real defects in pull requests that the Baseline ships.* Measured by `independent_test_defects` : the count and concrete evidence of adjudicated behavioral defects in Baseline pull requests exposed by running the Candidate's tests against the Baseline's code (`cross_tests`), where a cross-test failure alone is investigated rather than counted automatically [1].
- **H2 (Verification rounds and test volume).** *Verification rounds increase test code relative to the production change.* Measured per trial by `production_lines` (added + deleted lines outside `tests/`), `test_lines` (added + deleted lines under `tests/`), the `test_lines / production_lines` ratio, and `verification_rounds` [11, 12].
- **H3 (Planning questions and unattended stops).** *Open planning or setup questions stop or delay the Candidate's unattended trials.* Measured by collector trial outcomes (`delivered` , `stopped-to-ask` , `failed`) and

`planning_question_delay_minutes` (time spent waiting on or routing around unanswered questions, excluding provider usage-limit stalls).

- **H4 (Context re-reading and pre-production spend).** *Fresh roles re-read the same context repeatedly, and most of the Candidate's cost occurs before production code is edited.* Measured per trial by `repeated_context_reads` across fresh role sessions and `preproduction_usd / api_priced_usd` (the fraction of API-priced spend incurred before the first edit to a production file) [2, 10].
- **H5 (Fail-safe behaviour in unexpected cases).** *The Candidate's code fails safe more often in unexpected cases.* Using a fixed operational definition - code **fails safe** when, in a case it was not written for, it stops, refuses, reports an error or leaves data unchanged rather than sending, saving or showing something wrong - measured from the human grader's blind `unexpected_cases` assessments (`safe`, `unsafe`, `not_reached`, `unknown`) by counting `advantage_pairs` where one arm is `safe` and the other is `unsafe` on the same case.
- **H6 (Diff size, test coverage and grader preference).** *The human grader prefers smaller diffs, whereas model-based graders prefer larger test suites and higher coverage.* Measured by comparing each grader's and the author's pair preferences against each pull request's `diff_lines` (`lines_added + lines_removed`), `files`, `test_count`, `test_lines`, and measured `coverage_percent`.

Alongside H1–H6, we preregistered four methodological analyses:

1. **Position bias and order flips** for each of the eight model-based graders across `[X, Y]` and `[Y, X]` presentation orders [4, 5];
2. **Human–model agreement** (proportion and Cohen's κ with exact conservative simultaneous 95% confidence bounds) between the author and each model grader on the 17 first pairs;
3. **Self-preference and family bias**, comparing the full 6-grader panel majority against the 5-grader OpenAI-excluded panel (`sonnet`, `gemini`, `grok`, `muse`, `kimi`) and the 4-grader panel that excludes both Anthropic and OpenAI families (`gemini`, `grok`, `muse`, `kimi`), and comparing the same-family different-model panel graders (`claude-sonnet-5-5`, `gpt-6-astra`) against the exact-same-model diagnostic graders (`claude-opus-5-5`, `gpt-6.1-sol`) [8]; and
4. **Run-to-run repeatability** across the three seeded repeat tasks (`T8`, `T15`, `T17`), comparing how much each arm's outcomes, costs, times and grader preferences vary across two independent trials of the same task [9].

Section 2 describes the preregistered design, the two arms, task selection, isolation and grading apparatus. Section 3 documents the shakedown, the rubric pilot and the conduct of the twenty pairs. Section 4 reports the preregistered hypotheses (H1–H6), the grader and repeatability analyses, and exploratory findings from the transcripts. Section 5 discusses what the findings mean for `thirdshift`'s Ticket run and for multi-agent workflow design, and Section 6 assesses threats to validity.

2 Methods

2.1 Preregistration and terminology

The study design was worked out in a four-round structured design interview on 2–3 November 2026 and frozen in a private study repository before any counted pair was launched.

Following industry evaluation terminology [14], we use eight fixed terms throughout this paper (Appendix B):

- **Arm**: one of the two processes being compared, run whole with its orchestrator, roles, models, efforts and skills at pinned versions (the **Baseline** or the **Candidate**).
- **Task**: one closed Tourbot task with explicit acceptance criteria, given to both arms as the exact text it had when labelled `ready-for-agent`.

- **Trial**: one arm's attempt at one task, from the issue text to a pull request (`delivered`), an unattended pause for human input (`stopped-to-ask`), or an error (`failed`).
- **Pair**: the two trials on one task, one per arm, prepared and graded together.
- **Transcript**: everything a trial's sessions did - every message, tool call and result across the parent session and all child sub-agent sessions.
- **Outcome**: what a trial leaves behind (its pull request and repository tree, or its non-delivery state).
- **Grader**: whatever scores the outcomes of a pair - either a **model-based grader** (six **panel** graders and two **diagnostic** graders) or a **human grader** (the author across all delivered pairs, plus an optional colleague sample). We use **LLM-as-a-judge** only to name the evaluation method in the literature.
- **Fail safe**: when code meets a case it was not written for, it stops, refuses, reports an error or leaves data unchanged, rather than carrying on and sending, saving or showing something wrong.

2.2 Study design

The comparison is a preregistered, paired, blinded descriptive study across 17 tasks and 3 repeats (20 pairs, 40 trials). Each task is replayed from the commit immediately before its real fix was merged to `master` . On each pair, the Baseline and the Candidate run in parallel on separate, identical cloud machines, starting from the same commit and the same issue text. Neither arm has a separate pre-implementation specification stage: in Tourbot's workflow, issues labelled `ready-for-agent` already contain an implementation brief, so both arms begin directly from that issue text. All runs are unattended: no human answers questions during a trial, and a trial that stops to ask a question is recorded with outcome `stopped-to-ask` .

There is no adoption threshold, cost cap or sequential stopping rule. All 17 first-pair tasks form the primary descriptive population; the 3 repeated tasks are analysed separately for run-to-run consistency so that no task is double-counted in the 17-task summaries.

2.3 The two arms

Table 1. Configuration of the Baseline and Candidate arms as frozen on 3 November 2026.

Dimension	Baseline (`ticket-run`)	Candidate (Maker-Checker TDD)
Runner	`thirdshift 0.19.0`, Built-in Ticket run (`thirdshift --no-merge <issue>`)	`thirdshift 0.19.0`, Custom process (`thirdshift process <candidate> issue <issue>`)
Top-level orchestrator	Codex CLI `0.166.0`, `gpt-6.1-sol`, effort `high`	Codex CLI `0.166.0`, `gpt-6.1-sol`, effort `high` (Step 1: maker-checker TDD; Step 2: `delivery`)
Implementation & planning makers	Same single Codex `gpt-6.1-sol` (`high`) session	Separate fresh Codex `gpt-6.1-sol` (`high`) sessions (`test_planner`, `red_test_writer`, `green_implementer`, `refactorer`, `review_preparer`, `synthesizer`, `meta_reviewer`) via native Codex sub-agents or `codex exec`
Intermediate verifiers & fix checkers	None during implementation	Separate fresh sessions per stage (`test_plan_verifier`, `red_test_verifier`, `green_verifier`, `refactor_verifier`, `fix_checker`, `round_checker`); configured for `claude --print` (`claude-opus-5-5`, effort `medium`, Claude Code `2.1.312`) or independent sub-agent contexts per role execution rules
Code review stage	Built-in `spec_review` and `standards_review` (parallel Codex `gpt-6.1-sol` sub-agents) + `Security review` (`gpt-6.1-sol` session)	Cross-family review: parallel Claude (`claude-opus-5-5`, `medium`) and Codex (`gpt-6.1-sol`, `low`) reviewers across up to 5 rounds, plus bounded `meta-review` recovery routing
Unclear requirement or failed setup check	Session resolves by convention or assumption and continues	Escalation rules require pausing affected work and asking the human owner (`stopped-to-ask`)
Opening the pull request	Implementing session via `gh pr create`	`thirdshift` built-in `delivery` Step (`gpt-6.1-sol`, `high`)

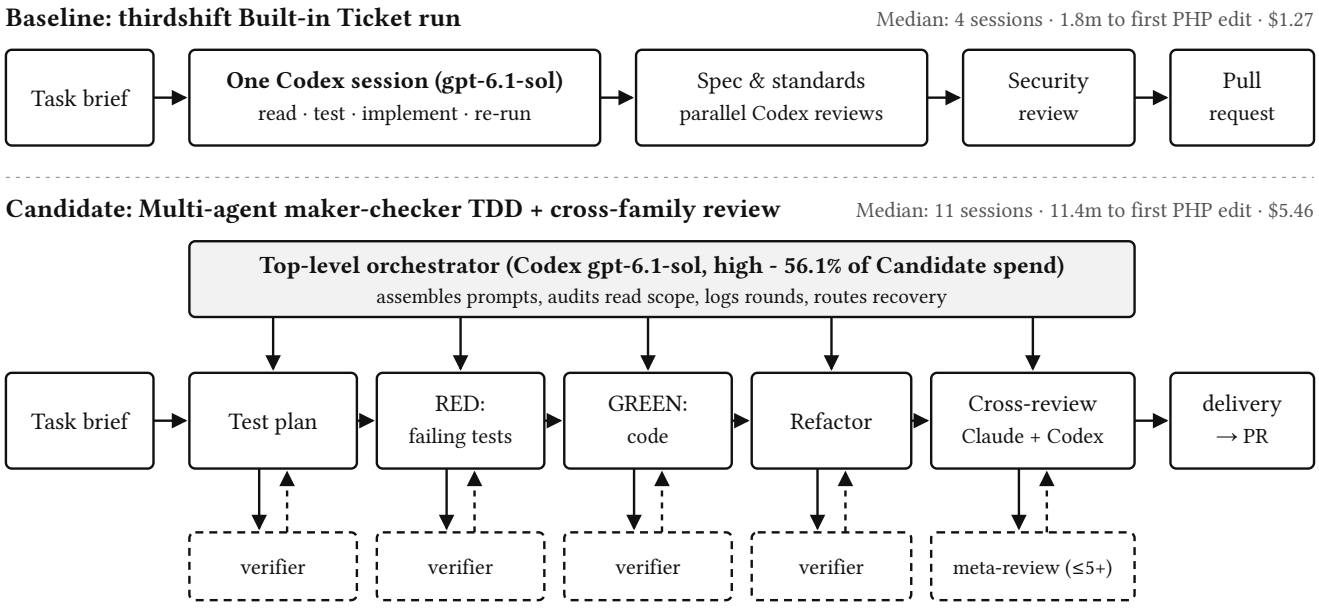


Figure 1. The two workflows side by side. The Baseline runs a single Codex session from brief to implementation and finishes with one parallel review pass and a security review. Under the Candidate, a top-level Codex orchestrator sequences distinct fresh maker and verifier contexts across planning, RED tests, GREEN implementation, refactoring, and multi-round cross-family review before handing off to `delivery`.

The key change from the first comparison is in the Candidate arm. In the first comparison, the Candidate's orchestrator ran on Claude Code (`claude-opus-5-5`, `medium`). Here, both arms use Codex `gpt-6.1-sol` at effort `high` as their top-level orchestrator, eliminating the orchestrator harness as a difference between the two arms.

We packaged the Candidate from its latest revisions frozen on 3 November 2026:

- the maker-checker TDD workflow (64 commits ahead of the first comparison's 29 October revision, adding explicit model-checking/experimentation rules, updated role execution, test planning, test verification, and workflow lifecycle guidance);
- the cross-family review workflow (revised 29 October 2026, adding round diagnoses and bounded meta-review routing); and
- supporting verification schemas, scripts, and specifications for Claude checker account routing.

Because the maker-checker TDD documentation at freeze time still noted that it did not yet supply concrete Claude CLI maker or intermediate-verifier recipes for a Codex orchestrator - and its proposed account-routing specification remained unimplemented - our packaging script recorded fourteen explicit interface gaps without inventing unshipped helper scripts. The frozen 47-file package checksum and process definition checksum were verified on the Candidate droplet before every pair.

2.4 Task selection, acceptance checklists and repeats

To fix the first comparison's unrecorded shortlist filter, we implemented the selection rule as a tested command-line script. Applied across Tourbot's closed tasks from 1 July to 3 November 2026, the script selected every task that:

1. was closed by a single merged pull request targeting `master`;
2. changed between 2 and 8 files and between +13 and +330 lines (additions + deletions), with no database migration or staging-only changes;
3. was not enrolled in the first comparison's ten tasks or its shakedown;
4. had been labelled `ready-for-agent` (so its pre-implementation task brief could be recovered from issue edit history); and
5. contained an explicit acceptance-criteria section in that as-labelled brief, confirmed by the author against a single stated rule: *the task states at least one observable result that a reviewer could check as met or not met.*

Twenty tasks met criteria 1–4. Three were excluded under criterion 5: one task never labelled `ready-for-agent`, one incident fix whose checklist was half live-production deployment steps, and one incident task containing only a single inline acceptance line (which was used instead as the uncounted shakedown task, `T0`). All **17 remaining tasks** (`T1` – `T17`) were enrolled in chronological order of their `ready-for-agent` labelling (Table 2), eliminating any random draw from the task pool.

To measure run-to-run variation, the selection script drew **3 of the 17 tasks** (`T17`, `T8`, `T15`) using Python's `random.Random(20261103)` and scheduled them as Pairs 18–20 (`T17-r2`, `T8-r2`, `T15-r2`) after the 17 first pairs.

Table 2. The 17 enrolled tasks (`T1` – `T17`) and 3 seeded repeats.

Pair(s)	Task	Criteria	Repeat?	Summary
1	`T1`	9	No	Staff picker: suppress untiered guides from new selection without unassigning existing ones
2	`T2`	5	No	Background alert jobs: read unified note records through a consolidated view query
3	`T3`	5	No	Web server access policy: block dot-paths and repository metadata files with HTTP 404
4	`T4`	3	No	Deployment script: verify and repair shared group-writable directory permissions on checkout and rollback
5	`T5`	4	No	Local Docker development: serve the Apache virtual host from `public/` with a startup guard
6	`T6`	2	No	Maintenance cleanup: retire two obsolete time-punch backfill scripts
7	`T7`	4	No	Reservation details form: save document-completion flags when toggled with no other field changes
8, 19	`T8`	5	Yes (`T8-r2`)	Reservation lifecycle: grant group-leader portal access at contract stage without requiring a prior agreement email
9	`T9`	4	No	Coverage backfill: exclude unparsable legacy files when running historical PHPUnit coverage snapshots
10	`T10`	10	No	Proposal deck editor: consolidate slide-override application behind one tested module interface (`applyDeckOverrides()`)
11	`T11`	3	No	Coverage backfill: translate legacy `<coverage>` filter blocks in historical `phpunit.xml` snapshots
12	`T12`	4	No	Legacy endpoint retirement: remove an unused unauthenticated email-helper endpoint
13	`T13`	4	No	Staff file download: handle blanked legacy user access tokens during download authorization
14	`T14`	4	No	Staff navigation: fix reservation link clicks stalling the side-menu loader on non-activated groups
15, 20	`T15`	5	Yes (`T15-r2`)	Daemon watchdog: distinguish planned deployment restarts from crash loops
16	`T16`	3	No	Legacy procedural view: apply contextual output escaping to a tab-selection parameter
17, 18	`T17`	3	Yes (`T17-r2`)	Legacy procedural script: parameterize record-ID SQL queries while preserving zero-padded numeric IDs

Before any counted pair ran, the author reviewed and froze the numbered acceptance checklist for each of the 17 tasks. Criteria that could only be checked against the live production environment or a pull-request description (which is stripped from the blind packet) were either dropped or rewritten into observable code/test checks - for example, dropping `T1` 's live-booking verification bullet, `T4` 's live production-host deployment bullet, `T9` and `T11` 's PR-body reporting clauses, and `T14` 's post-labelling PR screenshot requirement, and rewriting `T2` 's criterion 5 and `T13` 's criterion 2 to refer to automated tests and token-selection behaviour rather than production database IDs. Every refinement and its reason were committed before any pair launched.

2.5 Isolation, blinding and packet preparation

The physical isolation from the first comparison was preserved:

- Each arm ran in its own isolated OS user account with its own private GitHub replay repository, its own fine-grained GitHub token scoped only to that repository, and its own Claude and Codex subscriptions, on two separate 8-vCPU / 16 GB cloud droplets (Ubuntu 24.04).
- After every trial, an automated leak-search script scanned the full transcript (CLI JSONL streams, Codex rollouts, and child streams) for the original Tourbot issue number, closing PR number, branch name, commit SHA prefix, and distinctive lines of the real fix. Any coincidental match (such as token counts, JUnit line numbers, or an issue number already present in the parent commit or replay brief) was audited and recorded in private dismissal records.

- For each pair where both arms delivered a pull request, the packet builder randomly assigned the labels `X` and `Y` using a cryptographic shuffle (`random.SystemRandom().shuffle`), saving the mapping only in a `mode 600` operator file outside the grading environment.
- Inside the blind packet (`issue.md` , `checklist.json` , `checklist.md` , `pair.json` , `base/` , `X/` , `Y/` , `X.diff` , `Y.diff` , `evidence.json`), Git history, branch names, commit messages, pull request bodies, plan files and agent directories were stripped, and all changed `.php` files in `base/` , `X/` and `Y/` were formatted identically with `php-cs-fixer` (`v3.95.27` , `@PSR12`).

To give both model-based and human graders stronger execution evidence, packet preparation executed six automated verification runs per pair inside disposable copies under the unprivileged grading account:

1. **Cross-tests (`cross_tests.X_tests_on_Y` and `Y_tests_on_X`)**: each candidate's added/modified/deleted test files transplanted onto the other candidate's code and run through PHPUnit `10.5.63` (`Unit Tests` and headless Chrome `156.0.8078.79` `Browser Tests`).
2. **Base-tests (`base_tests.X_tests_on_base` and `Y_tests_on_base`)**: each candidate's test changes transplanted onto the unpatched `base/` tree, measuring directly whether each pull request's new tests actually fail without the production change.
3. **Mutation testing (`infection.X` and `infection.Y`)**: `infection/infection` (`0.35.8` with `PCOV 1.0.12`) scoped to the `src/**/*.php` files added or modified by that candidate relative to `base/` , reporting surviving and killed mutants with file and line numbers.

2.6 The structured rubric, grader panel and blind human grading

In place of the first comparison's single free-text `{winner, reason}` prompt, every model-based grader and the human grader evaluated each pair against the same five-part structured rubric (Appendix A):

1. **Acceptance criteria (`acceptance`)**: for each numbered criterion in `checklist.json` , score `X` and `Y` as `met` , `not_met` or `unknown` with concrete evidence (`evidence_X` , `evidence_Y`), and list any `other_defects` .
2. **Behaviour in unexpected cases (`unexpected_cases`)**: evaluate the same edge cases on both pull requests (missing/malformed data, database error or empty result, missing permission, encoding/collation differences), rating each arm `safe` , `unsafe` , `not_reached` or `unknown` , with `evidence_X` , `evidence_Y` and whether `executed` by a test.
3. **Scope and codebase fit (`not_required` , `fit` , `departures`)**: list any part of each diff that the task does not require (`harmless` , `review_or_maintenance_cost` , or `risk`) and rate fit with `base/` conventions (`follows` , `minor_departures` , `reshapes`).
4. **Tests (`tests_to_keep` , `tests_tied_to_implementation` , `untested_behaviour` , `cross_test_failures`)**: classify each cross-test failure as `defect_in_other_pr` , `test_tied_to_implementation` , `issue_ambiguous` or `unknown` , using `base_tests` and Infection surviving mutants.
5. **Merge-readiness and overall choice (`merge` , `changes` , `reason` , `winner` , `strength`)**: rate each candidate `as_is` , `small_fixes` or `no` with ordered changes required before merge, state the one-sentence main `reason` *before* choosing `winner` (`X` , `Y` , `tie`) and `strength` (`slight` , `clear`).

Table 3. The eight model-based graders and the blind human review assistant.

Slot	Role	Family	CLI & version	Model ID	Effort	Shares model with arm?	Shares family with arm?
`sonnet`	Panel (1/6)	Anthropic	Claude Code `2.1.312`	`claude-sonnet-5-5`	`max`	No	Yes (Candidate verifiers/reviewer)
`astra`	Panel (2/6)	OpenAI	Codex CLI `0.166.0`	`gpt-6-astra`	`ultra`	No	Yes (Both arms' `gpt-6.1-sol` roles)
`gemini`	Panel (3/6)	Google	`agy` `1.4.12`	`gemini-4-argon-high`	`high`	No	No
`grok`	Panel (4/6)	xAI	`grok` `1.0.70`	`grok-4.7`	`xhigh`	No	No
`muse`	Panel (5/6)	Muse	Muse Code `1.4.14`	`muse-wonder-full-1.0`	`xhigh`	No	No
`kimi`	Panel (6/6)	Moonshot	Claude Code `2.1.312` (Together AI)	`moonshotai/Kimi-K3.1-Pro[1m]`	`max`	No	No
`opus`	Diagnostic	Anthropic	Claude Code `2.1.312`	`claude-opus-5-5`	`max`	Yes (Candidate Claude roles)	Yes
`sol`	Diagnostic	OpenAI	Codex CLI `0.166.0`	`gpt-6.1-sol`	`ultra`	Yes (Both arms' Codex roles)	Yes
Review assistant	*Blind helper*	*Google*	`agy` *`1.4.12` (in grading env)*	*`gemini-4-argon-high`*	*`high`*	*No*	*No*

Each of the eight model-based graders was invoked twice per pair inside a read-only `bwrap` sandbox mounted at `/packet` - once with presentation order `["X", "Y"]` and once with `["Y", "X"]` (16 model calls per pair). An `X` choice scores `+1`, `Y` scores `-1`, and `tie` scores `0`; a grader's net preference is the sign of the sum across its two orders, so any order flip cancels to `0` (`tie`). Three panel views are reported:

1. **Six-LLM panel**: strict majority (≥ 4 of 6) across `sonnet`, `astra`, `gemini`, `grok`, `muse`, and `kimi`.
2. **Seven-voter combined panel**: strict majority (≥ 4 of 7) across the six LLM panel graders plus the human grader (author), giving an odd seven-voter panel spanning six model families and one human voter.
3. **Same-family-excluded panel**: strict majority across the panel graders that share no model family with either arm (or with the OpenAI-only arm when excluding OpenAI's `astra`, leaving an odd 5-LLM panel of `sonnet`, `gemini`, `grok`, `muse`, and `kimi`, or the 4 non-Anthropic/non-OpenAI panel graders `gemini`, `grok`, `muse`, and `kimi`). Diagnostic graders (`opus`, `sol`) never vote in the panel.

For **human grading**, the author opened each pair's blind review page, which displayed the task brief, the refined checklist, `X.diff`, `Y.diff`, and `evidence.json` (CI, cross-tests, base-tests, Infection), with no model grader outputs. To provide an uncontaminated baseline of human judgement, the review page locked the structured rubric form until the reviewer first submitted and sealed their **unaided merge call** (`merge as is`, `merge after small fixes`, or `don't merge`) for `X` and `Y`. Only after those initial calls were recorded could the author query the blind review assistant (`gemini-4-argon-high` running in the grading sandbox over the blind packet) and complete the full structured JSON form. Separate reviewer-scoped portals with isolated JSONL stores were also hosted for colleagues to record optional blind sample reviews on the same two-stage form.

2.7 Cost and time accounting per session and role

To test H3 and H4 and account for the Codex-orchestrated Candidate's mix of native Codex sub-agents (`spawn_agent`) and external `claude --print` invocations, the collection pipeline captured explicit parent and child session inventories alongside Codex session logs. Every parent stage, external CLI stream and native Codex rollout is matched to its role label from the invocation records, priced per model using a frozen published API price table (29 October 2026 rates), and aggregated into per-

session, per-role and trial-wide totals (`api_priced_usd`, `session_count`, `adjusted_minutes`, `outcome`). Any missing stream, unmatched rollout or unlabelled sub-agent marks the trial void.

3 Conduct of the comparison

3.1 Candidate shakedown and freeze

Before launching the 20 counted pairs, we ran one uncounted shakedown pair on 3 November 2026 on Tourbot shakedown task `T0` (legacy incident query parameterization) to verify the Codex-orchestrated Candidate package and the per-session metrics collector.

Both arms delivered green pull requests on `T0`. The Baseline completed in `9.99` minutes across `4` sessions (`implement`, `standards_review`, `spec_review`, `Security review`) at an API-priced cost of `$1.14` (`4.27 M` Codex input tokens, `23.5 k` output). The Candidate completed in `39.81` minutes across `13` sessions (orchestrator, `test_planner`, `plan_verifier`, `plan_verifier_v2`, `red_writer`, `red_verifier`, `green_maker`, `green_verifier`, `claude_review`, `codex_review`, `review_synthesis`, `final_review_verifier`, `final_review_verifier_v2`) at an API-priced cost of `$7.34` (`31.01 M` Codex input tokens, `121.0 k` output, `$6.21`; plus `1.16 M` Claude Opus 5.5 input tokens, `8.5 k` output, `$1.13`). Notably, switching the Candidate's orchestrator from Claude Code (first comparison) to Codex (`gpt-6.1-sol`) shifted 85% of the Candidate's API-priced spend on `T0` to OpenAI (`$6.21` of `$7.34`), with the top-level orchestrator session accounting for `$3.16` (43%) and the external Claude reviewer for `$1.13` (15%).

The `T0` shakedown exposed four operator/collector plumbing issues, all fixed before freezing:

1. the pair-launch script still pointed to the first comparison's package manifest rather than the updated v2 manifest;
2. under the Codex orchestrator, the Candidate workflow named its temporary scratch directory with a date-suffixed maker-checker path in `/tmp`, spawned its Codex roles as native sub-agents (`spawn_agent`), and wrote `exit.txt` rather than `exit-status.txt` in external Claude review directories;
3. on `T0` (a SQL-injection fix), the Baseline's `Security review` session triggered Codex's cybersecurity safeguard refusal before `codex exec --json` emitted a final `turn.completed` summary, requiring the collector to read parent token usage directly from the retained Codex rollout via the parent session inventory; and
4. `thirdshift 0.19.0` recorded `"outcome": "ready for review"` on `Run` commands and `"outcome": "succeeded"` on `Process` commands, both of which the metrics collector was updated to map to `"delivered"`.

After those fixes, both arms' collectors exited `0` with `void: false` and `warnings: []`, and the Candidate package was frozen before Pair 1 launched.

3.2 Rubric pilot and freeze

Before running any counted pair, we piloted the structured rubric under the unprivileged grading account against the three first-comparison pairs where the original two graders had disagreed (`Pilot P1`, `Pilot P2`, `Pilot P3`), running seven graders in both presentation orders and running `sonnet` a second time on each packet (`P1-repeat`, `P2-repeat`, `P3-repeat`) to test single-grader self-consistency.

Table 4. Pilot results across 5 panel graders, `sonnet` repeats, and 2 diagnostic graders on three first-comparison packets (blind labels `X` and `Y` preserved).

Packet	Panel majority	`sonnet` (run 1)	`sonnet` (run 2)	`astra`	`gemini`	`grok`	`muse`	`opus` (diag)	`sol` (diag)
`Pilot P1`	`Y` (5–0)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`0` (`X`/`Y`)	`−2` (`Y`/`Y`)
`Pilot P2`	`tie` (2 `Y`, 1 `X`, 2 `tie`)	`0` (`Y`/`X`)	`+2` (`X`/`X`)	`−2` (`Y`/`Y`)	`+2` (`X`/`X`)	`−2` (`Y`/`Y`)	`0` (`Y`/`X`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)
`Pilot P3`	`Y` (4 `Y`, 1 `tie`)	`0` (`X`/`Y`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`−2` (`Y`/`Y`)	`+2` (`X`/`X`)	`−2` (`Y`/`Y`)

The pilot established three points before freezing the rubric:

1. **Single-grader variance vs. panel stability.** On `Pilot P1`, `sonnet` was order-consistent in both runs (`−2`). On `Pilot P2` and `Pilot P3`, where `sonnet` rated both pull requests `small_fixes` with a `slight` margin, `sonnet` flipped with presentation order in run 1 (`net = 0`) and held one side in run 2 (`+2` on `P2`, `−2` on `P3`), while the 5-grader panel majority stayed stable (`tie` on `P2`, `Y` on `P3`). Swapping presentation order neutralizes single-run position flips on near-ties, and the multi-family panel majority absorbs single-grader variance on `slight` calls.
2. **Authoritative `checklist` numbering (Criterion 1).** On `Pilot P3`, the raw task brief had 6 bullets under acceptance criteria (including a manual sandbox timing note), whereas `checklist.json` had 5 numbered items. Before refinement, `grok` numbered `acceptance` against the 6 raw bullets on one order (failing schema validation) and penalized both candidates for the omitted manual timing note. Clarifying in Criterion 1 that `acceptance` must score `checklist`'s exact numbering and count - and that manual or PR-body notes omitted from `checklist` are outside the blind packet - resolved the failure (`grok` succeeded in both orders with `net = −2`).
3. **Explicit `base_tests` reference (Criterion 4).** We added `base_tests` explicitly to the packet contents list and Criterion 4 (`Tests`) of the rubric so graders would use the executed `X_tests_on_base` and `Y_tests_on_base` results rather than guessing whether tests would fail on `base/`.

We also fixed three CLI adapter details in the grading harness uncovered by the pilot: stripping `"$schema"` before passing the JSON schema to `claude --json-schema`, naming the temporary prompt file `prompt.txt` rather than `prompt.json` so `grok --prompt-file` would not parse it as an ACP wire message, and running the provider tracks concurrently while keeping `sonnet` → `opus` and `astra` → `sol` sequential on their shared provider logins. The rubric and schema were then frozen via SHA-256 verification before any counted run began (and `kimi` joined as the sixth panel grader for the counted study).

3.3 Flow of tasks and trials

All 17 replay branches and their 34 isolated replay issues across the two arm repositories were prepared and verified (`ci.conclusion == "success"`) before launching Pair 1.

During the first three pairs on 4 November 2026, three operational details in the metrics collector and scratch mirroring were updated to preserve complete evidence without modifying the frozen Candidate package:

1. **Mapping `No commits between` to `stopped-to-ask`.** On Pair 1 (`T1`), the Candidate completed its test plan and independent plan verification (`PASS`), then paused before RED/implementation to ask setup (`vendor/bin/phpunit` absent in an unbootstrapped worktree from a commit predating the repository's worktree-setup script), environment-guide, and review-sharing questions. Because the maker-checker TDD stage exited cleanly without making a commit, `thirdshift`'s subsequent `delivery` Step failed with `No commits between` the replay base branch and the task branch. We updated the collector so that when `delivery` fails with `No commits between`, the metrics collector records `--outcome stopped-to-ask`.
2. **Retaining Candidate scratch directories across locations.** Depending on how the Codex orchestrator interpreted its scratch path across trials, it wrote workflow artifacts either to `$worktree/.scratch/`, `$worktree/tests/_output/`, or

the harness's log directory. We updated our droplet mirror and collection script to retain all three locations before worktree cleanup.

3. **Bash array handling in the collector.** Fixed the shell wrapper to store command records and trace paths as bash arrays rather than newline-joined scalars when passing multiple `--session-trace` arguments.

All 20 pairs ran between 3 November and 5 November 2026 across the two isolated droplets. Every completed trial passed leak detection (`clean: true`), and every metrics record passed collector validation (`void: false`, `warnings: []`), so no pair was voided (`counted_pairs = 20`, `primary_pairs = 17`). All 13 pairs where both arms opened a pull request (`11` first pairs plus `2` repeats: Pair 18 `T17-r2` and Pair 20 `T15-r2`) were prepared and graded in both presentation orders (`[X, Y]` and `[Y, X]`) by the model graders inside `bwrap`. The author completed his blind human review of all 13 pairs on 6 November 2026 - sealing his unaided merge calls first and then his structured rubric judgements - before any model verdicts or X/Y mappings were opened. A colleague recorded independent blind unaided reviews in an isolated portal on Pair 10 (`T10`, 9 November 2026) and Pair 12 (`T12`, 16 November 2026) prior to opening any unblinded results.

Table 5. Overview of all 20 pairs (`17` primary tasks plus `3` seeded repeats): trial outcomes, API-priced cost, active minutes, session counts, production and test lines added (for delivered pairs), blind human merge calls, and 7-voter / 6-LLM panel winners.

Pair	Task	Baseline outcome, cost, time (sessions)	Candidate outcome, cost, time (sessions, verif. rounds)	Cost / time ratio (`C÷B`)	Prod / test lines (`B` vs `C`)	Author: `B` / `C` (`winner`)	7-voter / 6-LLM panel
1	`T1`	`delivered`, `$1.73`, `17.4m` (`4s`)	`stopped-to-ask`, `$1.99`, `14.5m` (`5s`, `2r`)	`1.15×` / `0.83×`	-	*ungraded* (**B** by delivery)	**B** / **B**
2	`T2`	`delivered`, `$1.27`, `13.3m` (`4s`)	`delivered`, `$6.20`, `60.7m` (`11s`, `8r`)	`4.89×` / `4.57×`	`20/192` vs `26/454`	`as_is` / `small_fixes` (**B**)	**B** / **B**
3	`T3`	`delivered`, `$1.25`, `15.7m` (`4s`)	`failed` (timeout), `$64.22`, `699.4m` (`64s`, `18r`)	`51.39×` / `44.62×`	-	*ungraded* (**B** by delivery)	**B** / **B**
4	`T4`	`delivered`, `$1.32`, `23.8m` (`4s`)	`delivered`, `$12.71`, `89.1m` (`15s`, `5r`)	`9.65×` / `3.74×`	`129/229` vs `104/542`	`as_is` / `small_fixes` (**B**)	**B** / **B**
5	`T5`	`delivered`, `$1.22`, `12.4m` (`4s`)	`delivered`, `$26.42`, `205.0m` (`41s`, `21r`)	`21.69×` / `16.56×`	`67/158` vs `114/1,073`	`as_is` / `no` (**B**)	**B** / `tie`
6	`T6`	`delivered`, `$0.65`, `6.5m` (`4s`)	`delivered`, `$2.70`, `28.7m` (`9s`, `2r`)	`4.19×` / `4.42×`	`626/24` vs `626/23`	`small_fixes` / `small_fixes` (`tie`)	**B** / **B**
7	`T7`	`delivered`, `$1.63`, `15.5m` (`4s`)	`delivered`, `$4.83`, `41.0m` (`10s`, `0r`)	`2.96×` / `2.65×`	`63/320` vs `48/485`	`as_is` / `small_fixes` (**B**)	**B** / `tie`
8	`T8`	`delivered`, `$1.67`, `18.0m` (`4s`)	`stopped-to-ask`, `$0.44`, `3.2m` (`3s`, `1r`)	`0.27×` / `0.18×`	-	*ungraded* (**B** by delivery)	**B** / **B**
9	`T9`	`delivered`, `$1.19`, `19.2m` (`4s`)	`failed` (workflow push), `$11.86`, `89.0m` (`17s`, `3r`)	`9.93×` / `4.64×`	-	*ungraded* (**B** by delivery)	**B** / **B**
10	`T10`	`delivered`, `$1.02`, `16.2m` (`4s`)	`delivered`, `$7.73`, `55.8m` (`13s`, `4r`)	`7.55×` / `3.44×`	`92/163` vs `97/354`	`as_is` / `small_fixes` (**B**; Colleague: `small_fixes` / `small_fixes`, `tie`)	**C** / **C**
11	`T11`	`delivered`, `$1.18`, `18.9m` (`4s`)	`delivered`, `$5.46`, `44.9m` (`8s`, `4r`)	`4.62×` / `2.37×`	`8/95` vs `4/155`	`as_is` / `as_is` (`tie`)	**B** / **B**
12	`T12`	`delivered` (CI red), `$1.15`, `20.6m` (`5s`)	`delivered` (CI red), `$4.62`, `50.7m` (`13s`, `6r`)	`4.01×` / `2.46×`	`17/133` vs `17/309`	`as_is` / `small_fixes` (**B**; Colleague: `as_is` / `small_fixes`, **B**)	**B** / `tie`
13	`T13`	`delivered`, `$1.36`, `21.7m` (`4s`)	`stopped-to-ask`, `$0.89`, `11.3m` (`3s`, `1r`)	`0.66×` / `0.52×`	-	*ungraded* (**B** by delivery)	**B** / **B**
14	`T14`	`delivered`, `$1.00`, `20.2m` (`4s`)	`delivered`, `$5.74`, `66.1m` (`12s`, `4r`)	`5.72×` / `3.28×`	`4/76` vs `4/164`	`as_is` / `as_is` (`tie`)	**B** / **B**
15	`T15`	`delivered`, `$1.70`, `29.1m` (`4s`)	`delivered`, `$16.50`, `156.9m` (`23s`, `16r`)	`9.71×` / `5.40×`	`158/239` vs `240/896`	`as_is` / `no` (**B**)	**B** / `tie`

Pair	Task	Baseline outcome, cost, time (sessions)	Candidate outcome, cost, time (sessions, verif. rounds)	Cost / time ratio (`C÷B`)	Prod / test lines (`B` vs `C`)	Author: `B` / `C` (`winner`)	7-voter / 6-LLM panel
16	`T16`	`delivered`, `$1.39`, `20.0m` (`4s`)	`stopped-to-ask`, `$1.35`, `15.4m` (`3s`, `1r`)	`0.97×` / `0.77×`	-	*ungraded* (**B** by delivery)	**B** / **B**
17	`T17`	`delivered`, `$1.77`, `23.0m` (`4s`)	`delivered`, `$3.39`, `55.1m` (`7s`, `3r`)	`1.91×` / `2.39×`	`77/203` vs `56/608`	`as_is` / `no` (**B**)	**B** / `tie`
18	`T17-r2` *(repeat)*	`delivered`, `$1.76`, `27.4m` (`4s`)	`delivered`, `$8.12`, `59.6m` (`11s`, `7r`)	`4.61×` / `2.18×`	`164/109` vs `22/365`	`as_is` / `small_fixes` (**B**)	**C** / **C**
19	`T8-r2` *(repeat)*	`stopped-to-ask`, `$0.13`, `2.4m` (`2s`)	`stopped-to-ask`, `$0.50`, `4.7m` (`3s`, `0r`)	`3.84×` / `1.99×`	-	*ungraded* (`tie`, neither delivered)	`tie` / `tie`
20	`T15-r2` *(repeat)*	`delivered` (CI poller EOF), `$1.54`, `16.2m` (`4s`)	`delivered`, `$13.15`, `121.8m` (`19s`, `14r`)	`8.51×` / `7.53×`	`145/309` vs `327/1,005`	`as_is` / `no` (**B**)	**B** / `tie`

4 Results

Across the 17 primary tasks (Pairs 1–17), the Baseline delivered a pull request on all **17 of 17** tasks (`16` with green CI; on Pair 12 `T12` both arms hit an unrelated pre-existing portal test failure), at a median cost of **$1.27** (total `$22.50`; `$25.93` across all 20 pairs) and a median active time of **18.9 minutes** (total `311.4` minutes across 17 pairs). The Candidate delivered a pull request on **11 of 17** primary tasks (`10` with green CI, plus Pair 12 `T12`), at a median cost of **$5.46** (total `$177.07` across 17 pairs; `$199.13` across all 20 pairs) and a median active time of **55.1 minutes** (total `1,686.7` minutes across 17 pairs). Across all 17 primary pairs, the median paired Candidate-to-Baseline ratio was **4.62×** for cost (`4.615`) and **3.28×** for time (`3.277`). When restricted to the 11 primary pairs where both arms delivered a pull request, the Candidate's median cost ratio was **4.89×** (`$5.74` vs `$1.22`, range `1.91×` – `21.69×`) and its median time ratio was **3.44×** (`55.8` vs `18.9` minutes, range `2.37×` – `16.56×`).

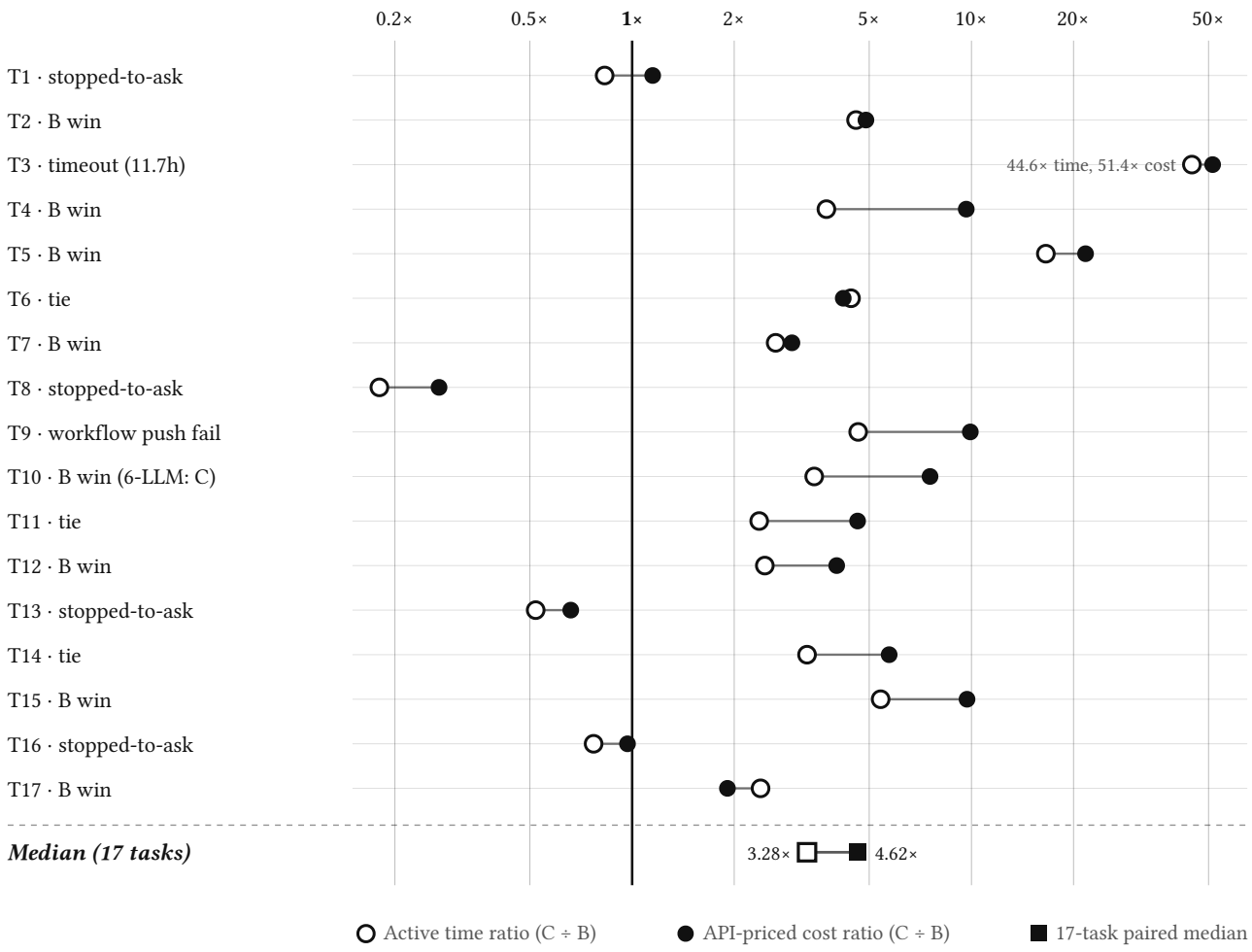


Figure 2. Candidate cost and active time as multiples of the Baseline across all 17 primary tasks (`T1` – `T17` , log scale). The solid vertical line marks equal cost or time (`1×`). Filled dots show the API-priced cost ratio and open dots show the active-time ratio; pairs to the left of `1×` (`T8` , `T13` , `T16`) exited early via `stopped-to-ask` without opening a pull request. Pair 3 (`T3`) timed out after `699.4` minutes (`51.39×` cost, `44.62×` time) without delivering a pull request.

4.1 Preregistered hypotheses (H1–H6)

- **H1 (Independent test fault detection - *not supported on these 11 delivered pairs*).** Across the 11 delivered primary pairs (`13` delivered pairs including repeats), the Candidate's independently authored tests (`candidate_tests_on_baseline`) detected **0** confirmed defects in any Baseline pull request (`independent_test_defects = 0`). The two cross-test failures of Candidate tests on Baseline code (Pair 5 `T5` and Pair 10 `T10`) both tested Candidate-specific internal helper structure rather than task requirements. Conversely, on Pair 17 (`T17`), a Baseline unit test checking zero-padded numeric string IDs (`'0010'`) failed on the Candidate's code (`baseline_tests_on_candidate`): the Candidate replaced the parent commit's `(int)` request-parameter cast with `filter_var($rawId, FILTER_VALIDATE_INT)` , which rejects numeric strings with leading zeros (`'0010'`) that `(int)` and `ctype_digit` accept.
- **H2 (Verification rounds and test volume - *supported*).** Across the 11 delivered primary pairs, the Candidate ran a median of **4.0** verification rounds per task (range `0` – `21` , total `91` rounds across the 17 primary tasks and `112` across all 20 pairs), compared with **1.0** review round for the Baseline. The Candidate added more test lines than the Baseline in 10 of the 11 delivered primary pairs (and tied at `23` vs `24` lines on the file-deletion task `T6`): a median of **454** test lines and **56** production lines per delivered task (**10.10** test lines per production line), against the Baseline's median of **163** test lines and

67 production lines (**2.64** test lines per production line), with a paired median test-line ratio of **2.32×** (`5,063` vs `1,832` total test lines across the 11 primary delivered pairs; `6,433` vs `2,250` across all 13 delivered pairs).

- **H3 (Planning questions and unattended stops - *supported*).** On the 17 primary tasks, the Candidate stopped without opening a pull request on **6 of 17** tasks (`35.3%`), whereas the Baseline delivered a pull request on all **17 of 17**:
  - **Four `stopped-to-ask` trials (`44.4` minutes total):** Pair 1 (`T1` , `14.5m` , `$1.99`), Pair 8 (`T8` , `3.2m` , `$0.44`), Pair 13 (`T13` , `11.3m` , `$0.89`), and Pair 16 (`T16` , `15.4m` , `$1.35`). On `T1` , the Candidate completed two test-plan verification rounds and stopped to ask how to bootstrap PHPUnit in a worktree predating the repository's setup script; on `T8` , `T13` and `T16` , its test-plan stage identified edge-case questions in the task brief and halted unattended execution under its rule requiring escalation of unanswered product questions.
  - **Two `failed` non-deliveries (`788.4` minutes, `$76.08` total):** On Pair 3 (`T3`), the Candidate finished the server access-control fix and 543 HTTP tests in `1h44m` (`$8.10`) but spent the next `9h55m` in 15 `meta-review` recovery loops (`64` sessions, `699.4m` , `$64.22`) until the 12-hour process timeout fired without a commit. On Pair 9 (`T9` , `89.0m` , `17` sessions, `$11.86`), a Candidate reviewer asked for CI enforcement of a repository hygiene check script, prompting the Candidate to edit a CI workflow file under `.github/workflows/` ; `delivery` 's `git push` was then rejected because the arm's fine-grained PAT intentionally lacked the `workflow` scope.
- **H4 (Context re-reading and pre-production spend - *supported*).** Across the 17 primary pairs, Candidate sessions executed **2,934** file-read/search tool calls (`3,355` across all 20 pairs) against the Baseline's **729** (`848` across all 20 pairs; paired median **2.65×**, total ratio **4.02×**). Because each planner, verifier and RED test-writer started in a fresh context before any implementation began, the Candidate reached its first PHP file edit at a median of **11.4 minutes** against **1.8 minutes** for the Baseline (paired median ratio **6.11×**). Across all 20 pairs, **76.0%** (`$150.32` of `$197.70` ; **74.5%** on the 17 primary pairs) of the Candidate's API-priced spend occurred in the orchestrator (`$110.87` , `56.1%`), test planning (`$24.59` , `12.4%`) and RED test writing (`$14.86` , `7.5%`), whereas green implementation accounted for only **$13.97** (`7.1%`) - almost identical to the Baseline's total implementation spend of **$14.53** across all 20 pairs.
- **H5 (Fail-safe behaviour in unexpected cases - *no net advantage*).** Under Criterion 2 of the rubric (`unexpected_cases`), neither arm held a net fail-safe advantage (`advantage_pairs = 0`). The human grader recorded `0` unexpected-case defects on either arm across all 13 graded pairs; among model graders, isolated unexpected-case notes split evenly between the two arms (such as Pair 17 `T17` 's leading-zero string ID handling favoring Baseline and Pair 15 `T15` 's non-blocking `flock` timeout favoring Candidate).
- **H6 (Diff size, test coverage and grader preference - *supported*).** On all **8 of 8** non-tied primary pairs (`10 of 10` including repeats), the blind human reviewer preferred the pull request with **fewer total diff lines and fewer test lines** (`lower = 8` , `higher = 0` , `tie = 3`). In his blind notes, the author cited excessive, verbose or over-engineered tests while rating below `merge as is` on the Candidate pull request on **10 of the 13** graded pairs (`T2` , `T4` , `T5` , `T7` , `T10` , `T12` , `T15` , `T17` , plus `T17-r2` and `T15-r2`). Among the three remaining tied pairs, `T11` noted only fixture naming, `T6` flagged an unnecessary 23–24 line deleted-file test on both arms equally (`merge after small fixes` each), and `T14` noted *"perhaps more test than needed, but acceptable"* while rating both arms `merge as is` . By contrast, several model-based graders rewarded the larger diff on a substantial fraction of pairs: `astra` and `grok` preferred the larger diff on `4 of 9` non-tied pairs (`lower = 5, higher = 4`), `kimi` on `3 of 8` (`lower = 5, higher = 3`), and `sol` on `3 of 9` (`lower = 6, higher = 3`).

4.2 Human and model-based grader preferences, position bias and family bias

Table 6. Per-pair preferences across the author's blind review, the four panel aggregations, the six panel graders, and the two same-model diagnostic graders on the 13 delivered pairs. **B** = Baseline, **C** = Candidate, `tie` = order flip or equal tie, *miss* = CLI/schema failure on one or both presentation orders.

Pair	Task	Author (`B` / `C` call)	Author winner	7-voter (`+Author`)	6-LLM (`all 6`)	5-LLM (`-astra`)	4-LLM (`-family`)	`sonnet`	`astra`	`gemini`	`grok`	`muse`	`kimi`	`opus` (diag)	`sol` (diag)
2	`T2`	`as_is` / `small_fixes`	**B**	**B**	**B**	**B**	`tie`	**B**	**B**	**B**	**B**	`tie`	`tie`	**B**	**B**
4	`T4`	`as_is` / `small_fixes`	**B**	**B**	**B**	**B**	**B**	`tie`	**B**	*miss*	**B**	**B**	**B**	`tie`	**B**
5	`T5`	`as_is` / `no`	**B**	**B**	`tie`	**B**	`tie`	**B**	**C**	**B**	**C**	**B**	`tie`	**B**	**C**
6	`T6`	`small_fixes` / `small_fixes`	`tie`	**B**	**B**	**B**	**B**	**B**	`tie`	**B**	**B**	**B**	**B**	**B**	`tie`
7	`T7`	`as_is` / `small_fixes`	**B**	**B**	`tie`	`tie`	`tie`	**B**	**B**	*miss*	*miss*	**B**	**C**	`tie`	**B**
10	`T10`	`as_is` / `small_fixes`	**B**	**C**	**C**	**C**	`tie`	**C**	**C**	**B**	**C**	`tie`	**C**	`tie`	**C**
11	`T11`	`as_is` / `as_is`	`tie`	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	`tie`	**B**	**B**	**B**
12	`T12`	`as_is` / `small_fixes`	**B**	**B**	`tie`	**B**	`tie`	**B**	**C**	**B**	*miss*	`tie`	**B**	**B**	**B**
14	`T14`	`as_is` / `as_is`	`tie`	**B**	**B**	**B**	**B**	**B**	**C**	**B**	**B**	**B**	`tie`	**B**	**C**
15	`T15`	`as_is` / `no`	**B**	**B**	`tie`	`tie`	`tie`	**B**	**B**	**C**	**C**	`tie`	**B**	**B**	**B**
17	`T17`	`as_is` / `no`	**B**	**B**	`tie`	**B**	**B**	`tie`	*miss*	*miss*	**B**	**B**	**B**	**B**	*miss*
18	`T17-r2`	`as_is` / `small_fixes`	**B**	**C**	**C**	**C**	`tie`	**C**	**C**	*miss*	*miss*	**C**	**C**	**C**	**C**
20	`T15-r2`	`as_is` / `no`	**B**	**B**	`tie`	**B**	`tie`	**B**	`tie`	*miss*	*miss*	**B**	**B**	**B**	`tie`
11 first	**B–C–tie**	`10/1/0` vs `2/6/3`	**8–0–3**	**10–1–0**	**5–1–5**	**8–1–2**	**5–0–6**	**8–1–2**	**5–4–1**	**7–1–0**	**5–3–0**	**6–0–5**	**5–2–4**	**7–0–4**	**6–3–1**

Human merge readiness vs. model panels. In his unaided merge calls - which were identical on every pair to his final rubric merge calls - the author judged **10 of 11** primary Baseline pull requests (`12 of 13` with repeats) ready to **`merge as is`** and **1 of 11** (`T6`) ready to **`merge after small fixes`**, with **0** `don't merge`. For the Candidate, he judged **2 of 11** (`T11`, `T14`) ready to **`merge as is`**, **6 of 11** (`7 of 13`) ready to **`merge after small fixes`**, and **3 of 11** (`4 of 13`: `T5`, `T15`, `T17`, `T15-r2`) **`don't merge`**. On the 3 pairs where both pull requests received the same merge category (`T6` `small_fixes` / `small_fixes`; `T11` and `T14` `as_is` / `as_is`), the author recorded `tie`, giving an **8–0–3** human preference for the Baseline on the 11 delivered first pairs (`10–0–3` across all 13 delivered pairs, and **14–0–3** across all 17 primary tasks when counting the 6 Candidate non-deliveries).

Why model graders preferred the Candidate on Pair 10 (`T10`) and Pair 18 (`T17-r2`), and what a colleague's blind sample reviews showed on Pair 10 and Pair 12. On Pair 10 (`T10`, `applyDeckOverrides()` proposal deck editor extraction), both arms shipped functionally identical production code (`92` vs `97` lines across the deck editor, export script, and glossary), so the entire disagreement turned on test structure and volume. Four panel graders (`sonnet`, `astra`, `grok`, `kimi`) and diagnostic `sol` preferred the Candidate because its dedicated unit-test file (`317` lines; `354` total test lines) added named `#[DataProvider]` cases covering nonblank string `'0'` text overrides and first-cover hero fallback ordering, whereas the author (`merge as is` for Baseline vs `merge after small fixes` for Candidate, noting that the Candidate's dedicated test file was a bit excessive) and `gemini` preferred the Baseline (`163` test lines) for replacing the source-string assertions directly inside the existing photo-picker and version test files without introducing a 125-case process-isolated matrix (`opus` scored both orders `tie` on offsetting test trade-offs). Notably, Pair 10 was the first of two pairs reviewed blind by a colleague on the engineering team (sealed 9 November 2026 without the optional second-stage detailed rubric form). Rating **both** pull requests **`merge after small fixes`** (`tie` on unaided merge readiness), the colleague's blind notes independently captured both halves of that trade-off: on the **Baseline**, *"stronger input check but tests are less organized in files"* (Baseline explicitly tested non-scalar `array`, `int`, `bool`, and `stdClass` values and forged context keys, but placed all `applyDeckOverrides()` tests inside the existing photo-picker test file alongside picker markup checks); on the **Candidate**, *"tests are set up in a way that easy to diagnose when they fail (separate files)"* (echoing `sonnet`'s praise for separating the unit tests from the existing picker and version test files, while still flagging small fixes before merge rather than awarding a win to the larger suite). On Pair 12 (`T12`, legacy unauthenticated email-helper retirement; sealed 16 November 2026 without written notes), where both arms shipped a 17-line production change returning HTTP `410` (`133` Baseline test lines vs `309` Candidate test lines across `6` verification rounds) and the 6-LLM panel split `tie` (`astra` voting Candidate while `sonnet`, `gemini`, and `kimi` voted

Baseline), the colleague independently rated the **Baseline** `merge as is` and the **Candidate** `merge after small fixes` (**Baseline preferred**) - identical to the author's blind unaided calls (`1` Baseline win, `0` Candidate wins, `1` tie across the two colleague-reviewed pairs, with `0` Candidate pull requests rated `merge as is`). On Pair 18 (`T17–r2`), where the Candidate's second run fixed the `FILTER_VALIDATE_INT` leading-zero bug from Pair 17 and trimmed its test suite from 608 lines to 365 lines, the model graders preferred the Candidate's in-place helper inside the legacy procedural script, whereas the author preferred the Baseline's extraction of a date-repairing loader helper into a dedicated loader module and still judged the Candidate's 365 test lines excessive for a 22-line change.

Table 7. Per-grader performance across the 11 delivered primary pairs: complete pairs (out of 11), presentation-order flips (out of complete pairs), any order changes (including `tie` to non-tie), first-position preference rate across individual calls, diff-size direction (`lower` vs `higher` diff lines preferred), and agreement with the blind human reviewer.

Grader	Role	Complete pairs	Order flips (`X↔Y`)	Order changes	First-position rate	Diff size (`lower` / `higher` / `tie`)	Matches with human (rate, 95% CI)	Cohen's κ (exact conservative 95% CI)
`sonnet` (`claude-sonnet-5-5`)	panel	`11 / 11`	`2` (`18%`)	`3`	`10 / 22` (`45.5%`)	`8 / 1 / 2`	`5 / 11` (`45.5%`, `[0.17, 0.77]`)	`−0.29` (`[−1.00, 0.83]`)
`astra` (`gpt-6-astra`)	panel	`10 / 11`	`0` (`0%`)	`3`	`8 / 21` (`38.1%`)	`5 / 4 / 1`	`5 / 10` (`50.0%`, `[0.19, 0.81]`)	`+0.19` (`[−1.00, 0.88]`)
`gemini` (`gemini-4-argon-high`)	panel	`8 / 11`	`0` (`0%`)	`0`	`9 / 19` (`47.4%`)	`7 / 1 / 0`	`4 / 8` (`50.0%`, `[0.16, 0.84]`)	`−0.10` (`[−1.00, 0.90]`)
`grok` (`grok-4.7`)	panel	`9 / 11`	`0` (`0%`)	`1`	`10 / 20` (`50.0%`)	`5 / 3 / 0` (1 equal)	`3 / 9` (`33.3%`, `[0.07, 0.70]`)	`−0.20` (`[−1.00, 0.79]`)
`muse` (`muse-wonder-full-1.0`)	panel	`11 / 11`	`5` (`45%`)	`6`	`9 / 22` (`40.9%`)	`6 / 0 / 5`	`5 / 11` (`45.5%`, `[0.17, 0.77]`)	`−0.14` (`[−1.00, 0.83]`)
`kimi` (`Kimi-K3.1-Pro[1m]`)	panel	`11 / 11`	`3` (`27%`)	`4`	`12 / 22` (`54.5%`)	`5 / 2 / 4`	`5 / 11` (`45.5%`, `[0.17, 0.77]`)	`−0.03` (`[−1.00, 0.83]`)
`opus` (`claude-opus-5-5`)	diagnostic	`11 / 11`	`2` (`18%`)	`2`	`10 / 22` (`45.5%`)	`7 / 0 / 4`	`5 / 11` (`45.5%`, `[0.17, 0.77]`)	`−0.38` (`[−1.00, 0.83]`)
`sol` (`gpt-6.1-sol`)	diagnostic	`10 / 11`	`0` (`0%`)	`2`	`7 / 21` (`33.3%`)	`6 / 3 / 1`	`6 / 10` (`60.0%`, `[0.26, 0.88]`)	`+0.27` (`[−1.00, 0.93]`)

Position bias and family bias. Across all eight graders, first-position preference rates stayed close to 50% (`33.3%` to `54.5%`), confirming that evaluating both presentation orders (`[X, Y]` and `[Y, X]`) effectively neutralized single-call position bias - particularly for `muse` (`5` order flips in 11 pairs) and `kimi` (`3` order flips), which frequently flipped on near-tie pairs and were thereby resolved to `tie`. On family bias, the two OpenAI graders (`astra` at `5–4–1` and `sol` at `6–3–1`) were the most favorable to the Candidate (`3–4` Candidate wins out of 10 complete primary pairs), whereas the two Anthropic graders (`sonnet` `8–1–2` and `opus` `7–0–4`) and Google's `gemini` (`7–1–0`) strongly favored the Baseline. Removing `astra` lowers the strict-majority threshold from `4 of 6` to `3 of 5` voters. Three primary ties (`T5`, `T12`, `T17`) become **Baseline wins**, yielding `8 Baseline, 1 Candidate, 2 ties` on the 11 delivered first pairs; the repeat `T15–r2` also changes to Baseline. These were not evenly split votes: Table 6 records `3B/2C/1 tie` on `T5`, `3B/1C/1 tie/1 missing` on `T12`, and `3B/0C/1 tie/2 missing` on both `T17` and `T15–r2`. Removing both `astra` and `sonnet` (`4–LLM –family`: `gemini`, `grok`, `muse`, `kimi`) yields **5 Baseline wins, 0 Candidate wins, and 6 ties** on the 11 delivered first pairs. Why did raw Cohen's κ between individual model graders and the author sit near zero (`−0.38` to `+0.27`) despite `45%`–`60%` raw agreement and overwhelming 7-voter panel agreement (`10 of 11`)? Two reasons: first, on `T6`, `T11` and `T14` the author scored `tie` (`merge as is` / `merge as is` or `small_fixes` / `small_fixes`) while nearly every model grader picked `Baseline` with a `slight` margin (`6` of the disagreement cells); second, because the author's marginal distribution on the 11 pairs had `0` Candidate wins (`8 Baseline, 3 tie`), chance-agreement correction in κ heavily penalizes any off-diagonal cell.

Recurring themes in the 6-LLM panel on Baseline and Candidate. Across all 145 completed evaluations by the six panel graders (`sonnet`, `astra`, `gemini`, `grok`, `muse`, `kimi`) on the 13 delivered pairs, the structured rubric fields and written `reason`s show consistent patterns in what the panel praised and criticized on each arm (Table 8):

Table 8. Aggregate rubric dimension counts and recurring qualitative themes across the 145 completed 6-LLM panel evaluations on the 13 delivered pairs.

Rubric dimension / theme	Baseline (`ticket-run`)	Candidate (Maker-Checker TDD)
Merge call (`as_is` / `small_fixes` / `no`)	`57` (`39.3%`) / `85` (`58.6%`) / `3` (`2.1%`)	`41` (`28.3%`) / `91` (`62.8%`) / `13` (`9.0%`)
Codebase fit (`follows` / `minor_departures` / `reshapes`)	`82` (`56.6%`) / `59` (`40.7%`) / `4` (`2.8%`)	`43` (`29.7%`) / `94` (`64.8%`) / `8` (`5.5%`)
Unrequested changes (`not_required`) with `review_or_maintenance_cost`	`115` **items** (`0.79` per call)	`323` **items** (`2.23` per call - **2.81**× Baseline)
Unrequested changes (`not_required`) with `risk` / `harmless`	`39` `risk` (`0.27`/call) / `163` `harmless`	`26` `risk` (`0.18`/call) / `177` `harmless`
Implementation-coupled tests (`tests_tied_to_implementation`)	`240` **items** (`1.66` per call)	`368` **items** (`2.54` per call - **1.53**× Baseline)
Primary panel praises	Compact tests executing real SQL/browser state (`Pairs 2, 6, 11, 12, 14, 15, 17, 20`); conformance to `src/` and PHPUnit conventions (`Pairs 5, 6, 11, 12, 14, 15, 20`); inode-pinned `sudo` safety (`Pair 4`), `NULL`-flag lock guard (`Pair 7`), and leading-zero ID preservation (`Pair 17`)	Dedicated case-named test file and `'0'`/first-cover boundary checks (`Pair 10`); `php://input` stream traps in endpoint retirement tests (`Pair 12`); non-blocking `flock` timeout with raw-count fallback (`Pair 15`); minimal in-place procedural edits (`Pairs 4, 7, 17, 18`)
Primary panel criticisms	Extracting a new helper module in legacy procedural code (`Pair 18`); unrequested booking-class property declarations (`Pair 7`); blocking `flock` without timeout (`Pair 15`)	Heavy, bespoke test harnesses and fake SQL parsers (`Pairs 2, 5, 10, 11, 12, 14, 15, 17, 20`); settings-path reshaping and test-hook env vars in production files (`Pair 5`, `Pair 15`); shell test outside PHPUnit (`Pair 6`)

- **What the LLM panel valued in Baseline:**
  1. *Compact, conventional tests that execute shipping code against real state rather than re-implementing logic in fakes (`Pairs 2, 6, 11, 12, 14, 15, 17, 20`).* On Pair 2 (`T2`), all six panel graders preferred Baseline (`6–0`) because its 192-line test ran the daemon's shipping SQL against an in-memory SQLite table, whereas Candidate built a 240-line custom harness that re-implemented the unified message view in a fake `query()` parser and pinned exact SQL strings. On Pair 6 (`T6`, `5–0–1`), Pair 11 (`T11`, `4–1–1`), Pair 12 (`T12`), and Pair 14 (`T14`, `4–1–1`), the panel similarly favored Baseline for placing small regression tests inside the repository's existing PHPUnit and Playwright suites rather than inventing standalone shell scripts or synthetic harnesses.
  2. *Conformance to existing `src/` and repository conventions (`follows = 56.6%` vs `29.7%` for Candidate).* On Pair 5 (`T5`), `sonnet`, `gemini`, `muse` and `kimi` rated Candidate `no` / `reshapes` and preferred Baseline (`clear` margin) because Baseline cleanly updated the Apache vhosts, sign-in alias, and startup guard with a focused real-Apache test, whereas Candidate buried its change under 1,073 lines of subprocess/trap tests and settings-path reshaping. On Pair 15 (`T15`) and Pair 20 (`T15-r2`), `sonnet`, `astra`, `grok`, `muse` and `kimi` credited Baseline for implementing the daemon restart counter as a small class under the application's `src/` daemon namespace in roughly one-third the code of Candidate.

3. *Edge-case safety on privileged repairs, locked flags, and legacy IDs (`Pairs 4, 7, 17, 20`).* On Pair 4 (`T4`, `5–1`), five panel graders preferred Baseline because it inode-pinned privileged directory repairs and covered file-removal rollbacks, whereas Candidate's `chmod g+rwX` left incident directories at `0775` and used pathname-based `sudo` vulnerable to symlink races. On Pair 7 (`T7`), `sonnet`, `astra` and `muse` noted that Baseline prevented locked users from registering spurious changes on `NULL`-default completion flags; on Pair 17 (`T17`), `sonnet`, `astra`, `grok`, `muse` and `kimi` preferred Baseline because it preserved zero-padded numeric IDs (`'0010'`) and bound repair dates in PDO.

- **What the LLM panel criticized in Baseline:**
  1. *Extracting new helper modules in legacy procedural code (`Pair 18 / T17–r2`).* Whereas the human maintainer praised Baseline's extraction of a date-repairing loader helper into a dedicated loader module as a deep module with a simple interface, all six panel graders penalized Baseline (`reshapes` / `minor_departures`) and preferred Candidate's minimal in-place edit inside the procedural script.
  2. *Unrequested class property declarations and blocking locks (`Pairs 7, 15`).* On Pair 7 (`T7`), `gemini` and `kimi` penalized Baseline for declaring properties on the booking record class (which followed the repository's coding standards but touched outside the completion-flag save flow); on Pair 15 (`T15`), `gemini`, `grok` and `muse` preferred Candidate's non-blocking lock timeout over Baseline's blocking `flock`.
- **What the LLM panel valued in Candidate:**
  1. *Dedicated boundary matrices and input-stream traps (`Pairs 10, 12, 15`).* On Pair 10 (`T10`, `4–1–1` for Candidate), `sonnet`, `astra`, `grok` and `kimi` preferred Candidate for its dedicated case-named unit-test suite covering nonblank string `'0'` and first-cover hero fallback ordering (a test-file separation that the colleague's blind sample review also credited). On Pair 12 (`T12`), `astra` and `muse` favored Candidate because its test trapped `php://input` in addition to `$_POST`. On Pair 15 (`T15`), `gemini`, `grok` and `muse` favored Candidate's non-blocking `flock` timeout and fallback to raw crash detection.
  2. *Localized in-place edits on procedural scripts (`Pairs 4, 7, 17, 18`).* When Candidate kept its production diff smaller than Baseline (`104` vs `129` lines on `T4`, `48` vs `63` on `T7`, `56` vs `77` on `T17`, `22` vs `164` on `T17–r2`), panel graders credited its narrower production surface area.
- **What the LLM panel criticized in Candidate:**
  1. *Disproportionate test harnesses and maintenance overhead (`323` `review_or_maintenance_cost` items vs `115` for Baseline; `2.81×`).* Even though model graders tolerated larger diffs more often than the human maintainer (H6), they still flagged 2.81× as many unrequested maintenance-cost items and 1.53× as many implementation-coupled tests (`368` vs `240`) on Candidate pull requests - repeatedly citing custom reflection/eval harnesses (`T2`, `T7`), test-only environment-variable hooks in production scripts (`T5`, `T15`, `T15–r2`), and 350–1,073-line test suites for small fixes.

4.3 Run-to-run repeatability on the three repeated tasks (`T8`, `T15`, `T17`)

Table 9. Comparison of the first trial and the independent repeat trial on the three seeded repeat tasks (`T17` Pairs 17 & 18, `T8` Pairs 8 & 19, and `T15` Pairs 15 & 20).

Task & Pairs	Arm	First trial: outcome, cost, time, sessions, diff (`prod/test`)	Repeat trial: outcome, cost, time, sessions, diff (`prod/test`)	First vs. repeat human call & 6-LLM panel
`T17` (Pairs 17 & 18)	Baseline	`delivered`, `$1.77`, `23.0m`, `4s`, `77/203`	`delivered`, `$1.76`, `27.4m`, `4s`, `164/109`	**Human:** `B` (`as_is` / `no`) → `B` (`as_is` / `small_fixes`) **6-LLM:** `tie` → `C`
`T17` (Pairs 17 & 18)	Candidate	`delivered`, `$3.39`, `55.1m`, `7s` (`3r`), `56/608`	`delivered`, `$8.12`, `59.6m`, `11s` (`7r`), `22/365`	*(Candidate fixed Pair 17's `'0010'` bug in Pair 18)*
`T8` (Pairs 8 & 19)	Baseline	`delivered`, `$1.67`, `18.0m`, `4s` (green CI)	`stopped-to-ask`, `$0.13`, `2.4m`, `2s`	**Human:** *ungraded* (`B` by delivery → `tie`) **6-LLM:** `B` → `tie`
`T8` (Pairs 8 & 19)	Candidate	`stopped-to-ask`, `$0.44`, `3.2m`, `3s` (`1r`)	`stopped-to-ask`, `$0.50`, `4.7m`, `3s` (`0r`)	*(Candidate stopped to ask on both runs)*
`T15` (Pairs 15 & 20)	Baseline	`delivered`, `$1.70`, `29.1m`, `4s`, `158/239`	`delivered` (poller EOF), `$1.54`, `16.2m`, `4s`, `145/309`	**Human:** `B` (`as_is` / `no`) → `B` (`as_is` / `no`) **6-LLM:** `tie` → `tie` (`-astra`: `tie` → `B`)
`T15` (Pairs 15 & 20)	Candidate	`delivered`, `$16.50`, `156.9m`, `23s` (`16r`), `240/896`	`delivered`, `$13.15`, `121.8m`, `19s` (`14r`), `327/1,005`	*(Both Candidate runs exceeded 2 hours and 890 test lines)*

The three repeated tasks show how much variance is intrinsic to each workflow versus the task itself:

1. **Task-characteristic persistence (`T15` and `T8`).** On `T15` (daemon restart accounting), both Candidate runs triggered heavy multi-round verification (`23` sessions / `16` rounds / `156.9m` / `$16.50` / `896` test lines in Pair 15; `19` sessions / `14` rounds / `121.8m` / `$13.15` / `1,005` test lines in Pair 20), and both received `don't merge` from the blind human reviewer against `merge as is` for both Baseline runs. On `T8` (reservation group-leader portal access), the Candidate stopped to ask a planning question within 3–5 minutes on both runs (`$0.44` and `$0.50`); the Baseline delivered a green pull request in Pair 8 (`18.0m`, `$1.67`) but stopped to ask a clarifying question on Pair 19 (`2.4m`, `$0.13`), the only `stopped-to-ask` among all 20 Baseline trials.
2. **Implementation variance on `T17` (Pairs 17 & 18).** On `T17` (invalid itinerary ID handling), the Baseline's two runs had nearly identical cost (`$1.77` vs `$1.76`) and sessions (`4` vs `4`), and both were judged `merge as is` by the author. The Candidate's first run (`Pair 17`, `$3.39`, `608` test lines) introduced the `FILTER_VALIDATE_INT` leading-zero regression caught by the Baseline's cross-test and was rated `don't merge`; its repeat run (`Pair 18`, `$8.12`, `365` test lines) avoided that bug and won the model panel, though the blind human reviewer still preferred the Baseline (`merge as is` vs `merge after small fixes`).

4.4 Exploratory findings from the transcripts

Table 10. Work per run across all 20 pairs: per-trial medians, 20-pair totals, and the median of the 20 paired Candidate ÷ Baseline ratios.

Measure per run (20 pairs)	Baseline median (total)	Candidate median (total)	Paired median ratio (`C÷B`)
Agent sessions	`4.0` (`79`)	`11.0` (`290`)	**2.67**×
Model turns	`17.5` (`328`)	`52.0` (`1,695`)	**3.20**×
Tool calls (all kinds)	`148.0` (`2,891`)	`383.5` (`10,901`)	**2.80**×
- Reading and searching	`45.0` (`848`)	`127.5` (`3,355`)	**2.65**× (`3.96×` total)
- File edits	`13.0` (`258`)	`21.0` (`562`)	**1.91**× (`2.18×` total)
- Test runs	`18.0` (`323`)	`31.5` (`702`)	**1.81**× (`2.17×` total)
- Lint and static checks	`5.0` (`115`)	`7.0` (`141`)	**1.00**× (`1.23×` total)
- `git` commands	`12.0` (`256`)	`24.0` (`691`)	**1.88**× (`2.70×` total)
- GitHub (`gh`)	`16.0` (`339`)	`2.0` (`107`)	**0.17**×
- Delegating / waiting on sub-agents	`8.5` (`180`)	`57.0` (`1,555`)	**8.20**× (`8.64×` total)
- Other shell / helper commands	`31.0` (`572`)	`118.5` (`3,788`)	**4.30**× (`6.62×` total)
Tool calls by role: Make / Check / Orchestrate	`88.5` / `59.0` / `0`	`94.0` / `65.0` / `237.5`	`1.24×` / `1.25×` / -
Minutes to first edit of a PHP file	`1.8m`	`11.4m`	**6.11**×

Table 11. Where the time, tool calls and API-priced cost went across all 20 pairs.

Arm & Phase	Sessions	Verifier sessions	Active minutes	Tool calls	API-priced cost (`$`, share)
Candidate (Maker-Checker TDD, 20 runs)	**290**	**123**	**2,958.7**	**10,901**	**$197.70** (`100%`)
- Orchestrator (top-level Codex session)	`23`	`0`	`1,778.2`	`5,638`	`$110.87` (`56.1%`)
- Test plan & plan verification	`65`	`26`	`402.6`	`1,429`	`$24.59` (`12.4%`)
- RED: failing tests & RED verification	`42`	`17`	`208.7`	`1,030`	`$14.86` (`7.5%`)
- GREEN: implementation & GREEN verification	`32`	`10`	`199.1`	`816`	`$13.97` (`7.1%`)
- Refactor	`5`	`0`	`7.9`	`65`	`$1.02` (`0.5%`)
- Code review & meta-review	`84`	`63`	`232.7`	`1,281`	`$23.12` (`11.7%`)
- Other (PR body, audit helpers, setup)	`39`	`7`	`129.5`	`642`	`$9.28` (`4.7%`)
Baseline (`ticket-run`, 20 runs)	**79**	**58**	**334.4**	**2,891**	**$25.93** (`100%`)
- Implementation session	`21`	`0`	`193.5`	`1,755`	`$14.53` (`56.0%`)
- Spec and standards reviewers (parallel)	`38`	`38`	`62.2`	`621`	`$6.82` (`26.3%`)
- Security review	`20`	`20`	`78.7`	`515`	`$4.59` (`17.7%`)

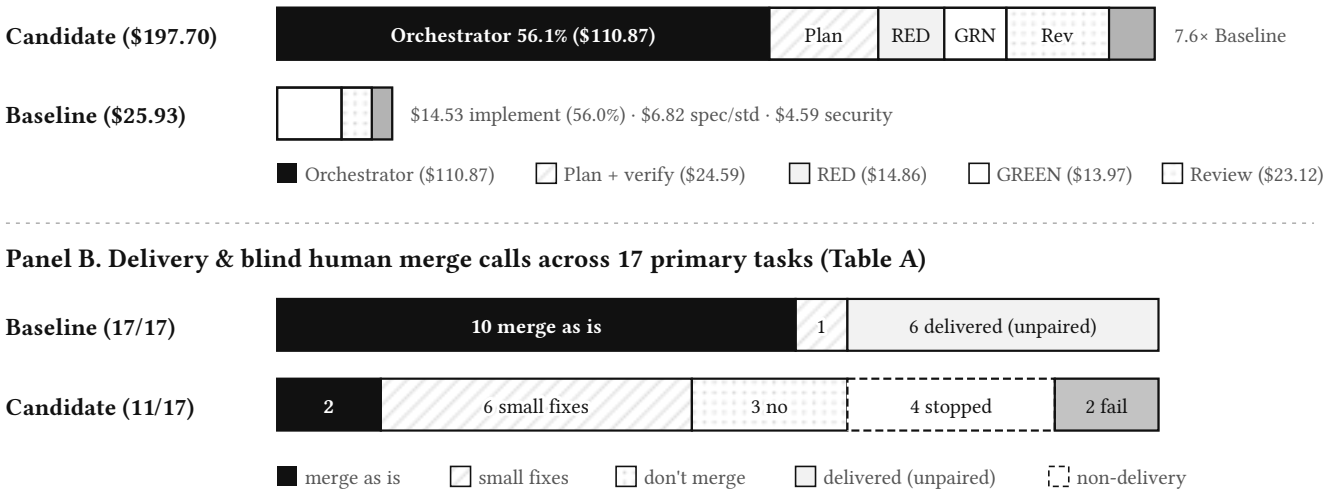


Figure 3. Spend allocation across all 20 pairs (Panel A) and primary-task delivery and blind human merge outcomes across the 17 tasks (Panel B). Over half (`56.1%`) of Candidate API-priced spend went to the top-level orchestrator alone, and green implementation spent `$13.97` (`7.1%`) - matching the Baseline's total implementation spend (`$14.53`).

1. **Why switching the Candidate orchestrator from Claude Code to Codex halved the cost ratio (`10.4× → 4.62×`) but made the orchestrator 56.1% of Candidate spend.** In the first comparison, Claude Opus 5.5 was the orchestrator and accounted for 85% of the Candidate's `$222.67` spend across 10 pairs (`$22.27`/pair). In this second comparison, running the orchestrator on Codex `gpt-6.1-sol` reduced the Candidate's average cost per pair to `$9.96` (`$199.13` across 20 pairs; median `$5.46` on the 17 primary pairs) because OpenAI's cached-input rate on `gpt-6.1-sol` (`$0.175/M`) is lower than Anthropic's cache rates. Even so, the Candidate's top-level orchestrator made **5,638 tool calls** (`51.7%` of all Candidate tool calls) and spent **$110.87** (`56.1%` of Candidate cost): after every sub-agent completed, the orchestrator hashed artifacts, ran read-scope audits over JSONL transcripts, updated verification and round logs, and assembled the next sub-agent's prompt. Notice that the actual `make` + `check` tool-call medians per run (`94.0` make and `65.0` check for Candidate vs `88.5` make and `59.0` check for Baseline) differed by only **1.24**× and **1.25**× - nearly the entire gap in tool calls came from the Candidate's **237.5** orchestration calls per run.
2. **Anatomy of an 11.7-hour verification spiral: Pair 3 (`T3`, web server access-control rule).** Task `T3` asked for a server access-control rule blocking direct HTTP requests to dot-paths, repository metadata, dependency manifests, and log/SQL dumps (`17` lines of shipping policy in total). The Baseline finished in `15.7` minutes (`$1.25`, 4 sessions) and shipped a clean rule and test. In the Candidate trial, the green implementer finished the server config fix and **543 passing HTTP policy assertions** against a real Apache container at **1 hour 44 minutes** (`$8.10`). At that point, the task was complete. However, during round 1 of cross-family review, a reviewer raised two findings on the integration test's local Apache process helper: `F1` (regex matching of the Apache port in foreground process cleanup) and `F2` (whether `stop()` could theoretically hit a PID-reuse window), even though the maker recommended treating `F2` as low non-blocking harness work and noted *"no collision frequency or failed canonical run is established"* (`FM-2.5`, *Ignored Other Agent's Input* [10]). Over the next **9 hours 55 minutes** (`52` additional sessions, `$56.12`), the interaction between the review stage's meta-review rules and the TDD stage's model-checking rules trapped the workflow in `meta1` through `meta11`:
   - The meta-review policy forbade deferring low-severity test-fixture findings to a follow-up issue (*"Don't create a follow-up issue or split the task to get under the round limit"*), and the round-limit rules allowed unlimited rounds past round 5 as long as each meta-review saw "changed evidence" (`FM-1.5`, *Unaware of Termination Conditions* [10]).
   - When the Candidate moved the test helper's teardown to an isolated process group (`setsid` + `posix_kill(-$pid, SIGTERM)`), subsequent reviewers asked for deterministic unit tests proving every branch of `stop()` (liveness, identity

mismatch, `SIGTERM` → `SIGKILL` escalation, and non-blocking `stream_select` readiness).

- When the RED verifier rejected seam-injected unit tests on the test helper because they failed for structural reasons on RED, `meta7` invoked the TDD skill's *"Model-check stateful or concurrent designs"* rule and ordered the planner to build a **32-row formal state-transition table (`T01` – `T32`) of the test fixture's `stop()` method**. By the end, its dedicated fixture-lifecycle test file had grown to **`1,101` lines and `41` cases (`31` failing, `10` passing)** for a **17-line** server config change, and four more hours (`meta8` – `meta11`) went into debating whether `T13` (`identity_check == 'unknown'`) was reachable after `T11` , until `thirdshift` 's 12-hour process timeout terminated the run at `699.4` minutes with no pull request opened.
- Across meta-reviews 12–14, the meta-reviewers explicitly diagnosed the churn in their own reasoning - invocation 12 wrote *"Twelve meta invocations and earlier capture/assignment failures are expensive churn,"* and invocation 14 wrote *"Thirteen earlier meta calls make another unchanged opinion or model loop unjustified"* - yet each still identified one more feasible correction and authorized another cycle (`FM-2.6` , *Reasoning–Action Mismatch* [10]).
- The exact same pattern recurred in miniature on Pair 5 (`T5` , `205.0m` , `41` sessions, `$26.42` , `1,073` test lines) and Pair 15 (`T15` , `156.9m` , `23` sessions, `$16.50` , `896` test lines), where rounds of review on test helpers forced repeated `plan → RED → GREEN` sub-cycles and produced test suites that caused the blind human reviewer to mark both pull requests `don't merge` .

5 Discussion

Table 12. Comparison of the first study (10 issues, Claude Code `claude-opus-5-5` orchestrator, 29 October rules) and this second study (17 primary tasks + 3 repeats, Codex `gpt-6.1-sol` orchestrator, 3 November rules).

Dimension	First comparison (`10` tasks, Claude orchestrator)	Second comparison (`17` primary tasks, Codex orchestrator)	Direction of change
Baseline delivery rate	`10 / 10` (`100%`)	`17 / 17` (`100%` ; `16` green CI)	Unchanged (`100%`)
Candidate delivery rate	`8 / 10` (`80%` ; `2` `stopped-to-ask`)	`11 / 17` (`64.7%` ; `4` `stopped-to-ask` , `2` `failed`)	Lower delivery (`-15.3 pp`)
Paired median cost ratio (`Candidate ÷ Baseline`)	**10.4**× (`$14.06` vs `$1.35`)	**4.62**× (`$5.46` vs `$1.27`)	Halved (`-56%` ratio)
Paired median time ratio (`Candidate ÷ Baseline`)	**6.4**× (`62m` vs `11m`)	**3.28**× (`55.1m` vs `18.9m`)	Halved (`-49%` ratio)
Median minutes to first PHP edit (`B` vs `C`)	`1.3m` vs `24.4m` (**21.4**×)	`1.8m` vs `11.4m` (**6.11**×)	Faster startup (`24.4m → 11.4m`)
Test lines per production line (`B` vs `C` , delivered)	`3.7` vs `14.7`	`2.64` vs `10.10` (medians)	Moderated (`14.7 → 10.10`), still `3.8×` Baseline
Blind human `merge as is` rate (delivered pairs)	**Baseline `7 / 8 ( 88% )`** vs **Candidate `1 / 8 ( 13% )`**	**Baseline `10 / 11 ( 91% ; 12/13` all)** vs **Candidate `2 / 11 ( 18% ; 2/13` all)**	Virtually identical (`~90%` vs `~15%`)
Blind human pair preference (delivered pairs)	**Baseline `6`** , Candidate `0` , Tie `2`	**Baseline `8`** , Candidate `0` , Tie `3` (`10-0-3` with repeats)	Identical pattern (`0` Candidate wins)
Independent cross-test defects found in Baseline (H1)	`2 / 8` delivered pairs (first comparison)	`0 / 11` delivered pairs (`1` Baseline cross-test caught Candidate `T17` bug)	Did not recur on the 17-task census

5.1 Why isolated maker-checker roles degraded: conceptual integrity, Cleanroom Goodharting, state machines and Shape Up

Five mechanisms established in software engineering and agent-systems research explain the empirical outcomes on H1–H6, complementing Ryan Singer's *Shape Up* framework [15]:

1. **Conceptual integrity and implicit context decisions (H4 - Brooks [16], Yan [17], Kim et al. [2]).** In *The Mythical Man-Month*, Brooks argues that conceptual integrity depends on keeping a coherent design in one mind rather than splitting tightly coupled implementation work across separate actors whose communication paths grow quadratically [16]. Yan's *Context-Sharing Principle* translates that rule to language-model agents: exploring a legacy codebase builds an implicit belief state in context, and every line of code or fixture setup carries implicit naming and structural decisions [17]. Splitting a 20-minute ticket across `11–64` fresh sessions forces `4.0×` as many file reads here (`5.2×` in the first study), delays the first code edit by `6.11×` (`21.4×` in Study 1), and causes isolated RED test writers to couple assertions to guessed internal helper names rather than task interfaces (`T5`, `T10`), mirroring multi-agent regressions reported across 260 configurations on SWE-bench Verified [2].
2. **Cleanroom test separation vs. Goodhart's Law on natural-language tickets (H2 - Mills et al. [18], Goodhart & Strathern [19], Hora & Robbes [11], Chen et al. [12]).** Separating test authorship from code implementation comes from safety-critical DO-178C and IBM Cleanroom engineering [18], where complete mathematical specifications define every signature and state transition before tests are written. On a 10–30 line natural-language issue in a legacy PHP monolith, an isolated RED writer cannot yet be graded on exposing bugs in shipping code; instead, survival depends on satisfying adversarial verifiers before implementation exists. As Goodhart's and Strathern's laws predict [19], verifier approval becomes the target: writers generate defensive mocks, source-file regex harnesses, and oversized suites (`14.7:1` test-to-code lines in Study 1; `10.10:1` here), consistent with findings that agent-generated suites lean heavily on mocks (`36%` vs `26%` [11]) and inflate cost and step counts without proportional resolution gains [12].
3. **Deterministic state machines vs. prose-interpreted loops (H3 - Schluntz & Zhang [20], MAST [10]).** Schluntz and Zhang distinguish deterministic code-orchestrated *workflows* from prompt-improvised *agents* [20]. `thirdshift`'s Built-in Ticket run fixes a 4-session Rust state machine with bounded retries (`17/17` delivered), whereas the Candidate asks an LLM orchestrator to interpret prose stop rules. The resulting non-deliveries map directly onto UC Berkeley's MAST failure taxonomy [10]: halting on minor ambiguities (`FM-3.1`, *Premature Termination*: `T1`, `T8`, `T13`, `T16`), disregarding pragmatic low-risk assessments (`FM-2.5`: `T3`), diagnosing futile churn while still authorizing another round (`FM-2.6`, *Reasoning–Action Mismatch*), and looping indefinitely wherever "changed evidence" exists (`FM-1.5`, *Unaware of Termination Conditions*).
4. **Intrinsic self-correction blindness vs. post-hoc diff-blind execution oracles (H1 - Huang et al. [21], Kamoi et al. [22], Knight & Leveson [23], Konstantinou et al. [1]).** LLMs cannot reliably self-correct without an external execution oracle [21, 22], and even separate reviewers anchored on the same diff exhibit correlated misses (`Knight & Leveson`, 1986 [23]) - explaining why diff-conditioned reviewers missed the coordinator-note deletion in Study 1 Pair 6 (`I6`) while transplanting an implementation-blind test caught it (`25%` vs `14%` detection [1]). Across all `19` unique delivered pairs in both studies (`8` in Study 1 + `11` here), however, Candidate cross-tests exposed confirmed Baseline defects on only `2 of 19` (`10.5%`: `I6` and `I7`, `0 of 11` in this 17-task census), while a Baseline unit test caught `1` Candidate regression (`T17`). Independent black-box probing pays when aimed selectively rather than applied indiscriminately to trivial deletions or config changes (`T6`, `T12`).
5. **Code-review attention economics (H6 - Sadowski et al. [24]) and *Shape Up* delivery guardrails (Singer [15]).** Google's empirical study of modern code review shows reviewer usefulness and defect detection degrade sharply as diff size climbs beyond small focal changes [24], explaining why the blind human maintainer preferred the smaller pull request on `8 of 8` non-tied pairs (`6 of 6` in Study 1) and flagged oversized Candidate test diffs (`10.10:1`) as a maintenance burden. Ryan Singer's *Shape Up* [15] turns those mechanisms into four concrete engineering rules: (a) **compare to baseline rather than an abstract ideal** (Chapter 14), rather than inventing theoretical edge cases; (b) **target the risk of not shipping and respect appetite** (Chapters 1–2), instead of spending 11.7 hours on a 17-line server rule; (c) **scope-hammer fixed time and variable scope** (Chapter 12), deferring low-severity fixture polish (`F2`) rather than forbidding follow-ups; and (d) **enforce hard circuit breakers** (Chapters 5 and 8), replacing open-ended meta-reviews with a fixed cap before escalation.

5.2 Which parts of the Candidate workflow are worth evaluating - and reconciling the "4+1 Blind Breaker" with H1 = 0

The objective of this second comparison was not to re-litigate wholesale adoption of the Candidate workflow, but to determine which specific ideas in it are worth testing inside `thirdshift`'s Built-in Ticket run - and how our earlier post-Study 1 "4+1 Hybrid" blueprint adjusts now that Study 2 returned `H1 = 0` (`2 of 19 = 10.5%` across both studies):

1. **Run on every ticket (`~$0`, seconds) - Deterministic `tests_on_base` verification and `assume_and_document` ambiguity handling.** Verifying that new regression tests fail against `base/` takes seconds inside the builder session via a deterministic command, and instructing unattended runs to document reasonable defaults in the pull request (`on_ambiguity = "assume_and_document"`) eliminates all four `stopped-to-ask` non-deliveries (`T1`, `T8`, `T13`, `T16`) at zero extra model cost.
2. **Evaluate selectively - A bounded `+1` Blind Behavioral Breaker fed `public_api_delta.txt` (`≤ 3` min, `≤ 150` test lines, `max_rounds = 1`).** Study 2's `H1 = 0` result across 11 small-to-medium tickets - alongside two cross-test false positives (`T5`, `T10`) caused by guessing private helper names - refines how an independent test pass should run in production. Rather than spending an extra test session on simple file deletions or config rules (`T6`, `T12`), a single cross-family breaker session (`claude-sonnet-5-5` or `claude-opus-5-5`) can be gated onto stateful data/text-mutation tickets (like Study 1's `I6` and `I7`) or onto the whole-feature (`Spec`) boundary where our earlier Fresh Eyes study showed a second look finds the highest yield. Giving the breaker only the issue, base repository, and a deterministically extracted public signature header (`public_api_delta.txt`, stripping all method bodies and `tests/` diffs) prevents helper-name mismatches while keeping the oracle blind to implementation logic [21]. Passing probes that add no branch coverage are discarded to keep PR diffs clean [24]; failing assertions resume the warm-context builder for at most **one** repair turn.
3. **Worth evaluating - A single, bounded cross-family code review pass (`max_rounds = 1`).** The Ticket run already spawns parallel `spec_review`, `standards_review`, and `security_review` sub-agents on OpenAI `gpt-6.1-sol` at `$0.57`/run. Our 8-grader panel data in Section 4.2 showed that non-OpenAI models (`sonnet`, `gemini`, `kimi`) caught test bloat and over-engineering that OpenAI models (`astra`, `sol`) tolerated. Adding **one** single-round non-OpenAI reviewer (`claude-sonnet-5-5` or `gemini-4-argon-high`) in parallel, capped at one repair turn, tests cross-family diversity without multi-round `meta-review` loops.
4. **Worth evaluating - Explicit test-proportionality, no-test-helper-spiral, and hard runner circuit-breaker rules.** Across both comparisons (`8/8` non-tied pairs here and `6/6` in Study 1), oversized test harnesses drove human rejections. Review prompts should flag source-string regex assertions, tests of deleted files (`T6`), and recursive unit-testing of test fixtures (`T3`), while loop limits and timeouts live in `thirdshift`'s Rust runner code rather than in agent-interpreted prose.
5. **Do not adopt - Multi-session pre-implementation role splitting (`plan → tpv → red → redv → green → greenv`) and open-ended `meta-review` routing.** Splitting a 20-minute ticket across `11-64` fresh sessions multiplied orchestration spend (`56.1%` of Candidate cost), delayed first code edits by `6.11×`, caused `6/17` unattended non-deliveries, and produced `0` cross-test defect wins across the 17-task census.

6 Threats to validity

Construct validity.

- *Structured rubric and grader preferences.* Unlike the first comparison, every model-based grader and the human grader filled in the same structured fields (`acceptance`, `unexpected_cases`, `not_required`, `fit`, `tests`, `merge`, `winner`, `strength`) before stating a preference, and four of the six panel graders (`gemini-4-argon-high`, `grok-4.7`, `muse-wonder-full-1.0`, `moonshotai/Kimi-K3.1-Pro[1m]`) come from model families that neither arm uses. We also report the two exact-model diagnostic graders (`claude-opus-5-5`, `gpt-6.1-sol`) separately. Even so, model-based graders can

share general LLM priors (such as favouring longer diffs or more defensive branches; H6), which is why human judgement carries primary weight.

- *Base-tests and mutation testing.* Adding `base_tests` (`X_tests_on_base`, `Y_tests_on_base`) directly measures whether new tests fail on the unpatched codebase. Infection mutation testing remains scoped to changed `src/**/*.php` classes; tasks whose production changes lie in legacy procedural scripts, helper scripts, server config, Docker, or `phpunit.xml` produce `0` mutations from Infection, so test-quality grading on those tasks relies on `base_tests`, `cross_tests` and code inspection.
- *API-priced cost.* As in the first comparison, pricing tokens at frozen published API rates provides a single unit to sum OpenAI and Anthropic usage across sessions and roles, rather than a literal subscription invoice.

Internal validity.

- *Blinding and test volume cues.* All Git metadata, branch names, commit messages, PR descriptions and process files were stripped from `/packet`, and changed PHP files were formatted with `php-cs-fixer`. However, as in the first comparison, a pull request with a very large test suite can still allow a human or model grader to guess which arm produced it. Recording every criterion in the structured rubric before `winner` makes any size preference visible and testable under H6.
- *Blind human review assistant.* To prevent the author's interactive review helper from biasing his initial merge readiness judgement, the review workflow enforced a two-step lock: the author recorded and sealed his unaided merge calls (`X` and `Y`) before optionally consulting the `gemini-4-argon-high` helper in the grading sandbox or completing the detailed structured rubric form. In practice, the author's unaided merge calls were identical to his final rubric merge calls on all 13 graded pairs.
- *Process fidelity and the Codex-orchestrated reconstruction.* We ran the Candidate on its chosen top-level harness and model (Codex `gpt-6.1-sol`, `high`) and its latest skill commits as of 3 November. However, because its Codex-to-Claude account-routing specification remained unimplemented at freeze time, the orchestrator invoked Claude and Codex sub-agents using the skill's general role execution rules. In practice, the Codex orchestrator frequently spawned Codex roles via native `spawn_agent` threads in addition to `claude --print` calls.
- *Historical repository state on older tasks.* Four of the seventeen tasks (`T1`, `T3`, `T4`, `T6`) replay parent commits from September 2026, before a unified worktree-setup script was added to Tourbot (the remaining thirteen replay commits from 15–30 October 2026 include that setup script). Both arms faced the exact same parent tree on every task, but the Candidate's rule against running unverified setup commands (`"Don't run setup automatically to repair a failed check"`) interacted differently with those four older commits (causing `T1` to pause with `stopped-to-ask`, while `T3` ran `composer install` and proceeded).
- *Limited multi-reviewer human sample.* Because other team members were at capacity on feature work, the author was the only human grader to score all 13 delivered pairs; one colleague completed independent blind unaided reviews on two pairs - Pair 10 (`T10`, rating both pull requests `merge after small fixes`, `tie`) and Pair 12 (`T12`, rating the Baseline `merge as is` and the Candidate `merge after small fixes`, matching the author's call; Section 4.2) - and a second colleague's review portal received no submissions prior to final publication.

External validity.

- *Task scope and starting point.* Although expanding from 10 to 17 tasks (plus 3 repeats) covers the entire eligible pool of closed `ready-for-agent` Tourbot tasks with acceptance criteria through 3 November 2026, all 17 tasks are still small-to-medium changes (2–8 files, +13 to +330 lines in the shipped fix) to one PHP application, starting from an already-triaged `ready-for-agent` brief. Neither comparison tests greenfield feature development or upstream specification shaping.

Conclusion validity.

- *Sample size and conservative intervals.* Seventeen first-pair tasks (plus three repeats) provide a complete census of eligible Tourbot tasks in the study window and narrow binomial intervals by roughly a quarter compared with ten tasks [7], while the three repeated tasks directly bound run-to-run noise on the same task. Our simultaneous Bonferroni–Clopper–Pearson confidence enclosures for Cohen's κ make no distributional or independence assumptions, though they are intentionally wide at $N = 11$ delivered primary pairs.

7 Conclusion

Across 20 preregistered, blinded replay pairs (`17` primary tasks plus `3` repeats), running the updated multi-agent maker-checker TDD workflow under a Codex `gpt-6.1-sol` (`high`) orchestrator cut the Candidate-to-Baseline cost and time ratios roughly in half compared with the first study (`4.62×` cost and `3.28×` time vs `10.4×` and `6.4×`), yet the single-agent Built-in Ticket run (`ticket-run`) remained substantially more reliable, faster, cheaper and more merge-ready:

1. **Delivery (H3).** The Baseline delivered a pull request on **17 of 17** primary tasks (`16` with green CI); the Candidate delivered on **11 of 17** (`10` with green CI), stopping to ask planning/setup questions on 4 tasks (`T1` , `T8` , `T13` , `T16`), timing out after 11 hours 39 minutes (`64` sessions, `$64.22`) of recursive test-fixture meta-review on 1 (`T3`), and failing delivery after editing a CI workflow file on 1 (`T9`).
2. **Merge readiness and human preference (H6).** On the 11 delivered primary pairs (`13` with repeats), the blind human maintainer rated **10 of 11** (`12 of 13`) Baseline pull requests **`merge as is`** (`0` `don't merge`), compared with **2 of 11** (`2 of 13`) **`merge as is`** and **3 of 11** (`4 of 13`) **`don't merge`** for the Candidate - preferring the Baseline on **8 of 11** delivered primary pairs (`3` ties, `0` Candidate wins; `14–0–3` across all 17 primary tasks) and choosing the smaller diff on **8 of 8** non-tied pairs.
3. **Model panel consensus and family bias.** The 7-voter panel (`6 LLMs + author`) preferred the Baseline on **10 of 11** delivered primary pairs (`1` Candidate win on `T10` , `0` ties; `16 of 17` overall), and the 5-LLM OpenAI-excluded panel preferred the Baseline on **8 of 11** (`1` Candidate win, `2` ties).
4. **Where the work went (H1, H2, H4).** Candidate cross-tests detected **0** confirmed defects in Baseline code across the 11 delivered primary pairs (while a Baseline cross-test caught 1 regression in Candidate `T17`), while the Candidate spent **76.0%** of its 20-pair cost in orchestration (`56.1%`), planning (`12.4%`) and RED test writing (`7.5%`) and generated a median **10.10** test lines per production line against **2.64** for the Baseline.

ETA will therefore continue using `thirdshift` 's single-agent Ticket run for Tourbot production work (including upcoming multi-ticket feature builds) and evaluate only bounded, single-pass mechanisms - automated `tests_on_base` checks, a single parallel cross-family review pass, and explicit test-proportionality and circuit-breaker rules - in future iterations of `thirdshift` .

8 Declarations

Data and code availability. `thirdshift` is open source at thirdshift.app and github.com/JacobStephens2/thirdshift. Every aggregated metric, per-pair outcome, diff-size measurement, and human and model-grader call from this study is reported in Tables A and 1–12 of this paper (and in condensed machine-readable form in `agents.md`). The replay repositories, raw agent transcripts, per-task diffs, and Candidate skill files are withheld because they contain private Tourbot production application code and third-party workflow materials.

Conflicts of interest. As in the first comparison, the author leads engineering on Tourbot at Educational Travel Adventures, commissioned both comparisons, graded the pull requests blind under a two-stage lock before unblinding, and develops `thirdshift` (which executed both arms and provides the Built-in Ticket run). The Candidate's maker-checker TDD and cross-family review workflow was authored independently and proposed for adoption on Tourbot.

Funding. Internal; model usage ran on Educational Travel Adventures' provider subscriptions and cloud droplets.

AI use and contributions. The author shaped and approved the study design, refined and froze the 17 task checklists, and recorded blind human judgements across all 13 delivered pairs; a colleague recorded independent blind sample reviews on Pair 10 (`T10`) and Pair 12 (`T12`). Experiment tooling and orchestration were built and run with AI coding agents (`gemini-4-argon` / Antigravity and `thirdshift` runs), which also drafted this manuscript and its public adaptation for the author's review. All methodology decisions and reported judgements are the author's.

References

1. Konstantinou, M., Tambon, F., and Papadakis, M. On the risk of coding before testing: An empirical study on LLM-based test generation workflow. arXiv:2607.05139, July 2026. Tests generated after faulty code detected 14% of faults; tests generated independently, 25%.
2. Kim, Y., Gu, K., Park, C., et al. Towards a Science of Scaling Agent Systems. arXiv:2512.08296, December 2025. 260 agent-system configurations; on SWE-bench Verified every multi-agent architecture did slightly worse than a single agent (−2.1% to −14.9%).
3. Xiang, Z., Zhang, Y., Zhang, Y., and Xu, H. Cross-Model LLM Code Review: Should you use Claude to review Codex or vice versa? *Agentic SE @ KDD '26*, arXiv:2607.21656, July 2026. Claude drafts passed 91.4% alone and 82.8% after GPT review; GPT/Codex drafts passed 71.6% alone and 89.7% after Claude review.
4. Zheng et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *NeurIPS*, 2023. arXiv:2306.05685.
5. Shi et al. Position bias in pairwise LLM judging. *AACL*, 2025. Bias is strongest when candidates are close in quality.
6. Crupi et al. LLM-as-a-judge for code generation. *IEEE Transactions on Software Engineering*. arXiv:2507.16587. Reference-free judging agreed with execution results at Cohen's κ 0.10–0.21.
7. Miller. Adding error bars to evals. arXiv:2411.00640, 2024.
8. Pombal, J., Rei, R., and Martins, A. F. T. Self-Preference Bias in Rubric-Based Evaluation of Large Language Models. arXiv:2604.06996, April 2026. Judges passed failing tests far more often on code from their own model family.
9. Bjarnason, B. H., Silva, A., and Monperrus, M. On Randomness in Agentic Evals. arXiv:2602.07150, February 2026. Single-run pass@1 estimates vary by 2.2–6.0 points.
10. Cemri et al. Why do multi-agent LLM systems fail? arXiv:2503.13657, 2025. The MAST taxonomy: 14 failure modes in three categories (specification or system design, inter-agent misalignment, task verification), from traces of seven multi-agent frameworks; inter-annotator Cohen's κ 0.88.
11. Hora and Robbes. Are coding agents generating over-mocked tests? An empirical study. *MSR*, 2026. arXiv:2602.00409.
12. Chen et al. Rethinking the value of agent-generated tests for LLM-based software engineering agents. arXiv:2602.07900, 2026.
13. Wan et al. From runnable to shippable: multi-agent test-driven development for generating full-stack web applications from requirements. *ASE*, 2026. arXiv:2605.17242.
14. Anthropic. Demystifying evals for AI agents. January 2026. (Source for task, trial, transcript, outcome and grader vocabulary.)
15. Singer, R. *Shape Up: Stop Running in Circles and Ship Work that Matters*. Basecamp, 2019 (chs. 1, 2, 5, 8, 12, 14). https://basecamp.com/shapeup.
16. Brooks, F. P., Jr. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1975 (chs. 2–4); and "No Silver Bullet - Essence and Accidents of Software Engineering," *Information Processing 86*, pp. 1069–1076, 1986.
17. Yan, W. Don't build multi-agents: principles of context engineering. Cognition AI Engineering Blog, June 2025.
18. Mills, H. D., Dyer, M., and Linger, R. C. Cleanroom software engineering. *IEEE Software*, 4(5):19–25, 1987.
19. Goodhart, C. A. E. Problems of monetary management: the U.K. experience. *Papers in Monetary Economics*, vol. 1, Reserve Bank of Australia, 1975; and Strathern, M., "'Improving ratings': audit in the British university system," *European Review*, 5(3):305–321, 1997.
20. Schluntz, E., and Zhang, B. Building effective agents. Anthropic Research & Engineering, December 2024.
21. Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. *ICLR*, 2024. arXiv:2310.01798.

22. Kamoi, R., Zhang, Y., Zhang, N., Han, J., and Zhang, R. When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. arXiv:2406.01297.

23. Knight, J. C., and Leveson, N. G. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1):96–109, 1986.

24. Sadowski, C., Söderberg, E., Church, L., Sipko, M., and Bacchelli, A. Modern code review: a case study at Google. *ICSE-SEIP*, pp. 181–190, 2018.

Appendix A Protocol and structured rubric summary

Replicating this comparison on another codebase requires six deterministic stages and a single shared grading schema:

A.1 Six-stage replay and blinding pipeline

1. **Deterministic task selection and checklist freeze.** Filter closed tasks by single-PR merge, file/line bounds, absence of database migrations or live-only dependencies, and presence of an explicit pre-implementation acceptance-criteria section; draw any repeatability sample using a recorded PRNG seed (`20261103` here). Freeze each task's numbered `checklist.json` before running any trial.
2. **Version and price-table freeze.** Pin the runner release, agent CLI versions, model IDs, reasoning efforts, skill revisions, and per-token API pricing table before the first pair runs.
3. **Parallel isolated replay execution.** Prepare identical replay branches at the parent commit immediately before each historical fix, create identical replay issues in isolated per-arm repositories on separate machines with repository-scoped credentials, and launch both arms unattended in parallel. Map clean exits with no commits (`No commits between` base and task branch) to `stopped-to-ask`.
4. **Per-session metrics collection and leak detection.** Retain all parent CLI logs, native sub-agent rollouts, and external child CLI streams; reject any trial with unmatched streams (`void: true`); and scan every transcript against the real closing PR, commit SHA, branch name, and fix diff lines (`clean: true`).
5. **Blind packet preparation and automated execution evidence.** Format all changed PHP files identically (`php-cs-fixer`), strip Git history, commit messages, PR bodies, and agent scratch directories, assign `X`/`Y` labels via `random.SystemRandom().shuffle` (storing the mapping in a `mode 600` file outside the grading sandbox), and execute six checks under an unprivileged grading account: `cross_tests` (`X` tests on `Y`, `Y` tests on `X`), `base_tests` (`X` and `Y` tests on `base/`), and scoped `infection` mutation testing on `src/`.
6. **Two-stage blind human grading and dual-order `bwrap` model grading.** Run each model grader twice per pair inside a read-only `bwrap` sandbox mounted at `/packet` (`[X, Y]` and `[Y, X]`), neutralizing single-call position flips to `tie`. Record and seal the human reviewer's unaided merge calls before unlocking the interactive blind review assistant or viewing any model verdicts.

A.2 Five-criterion structured rubric schema

Every human and model grader fills out the same JSON structure in order, recording concrete evidence before ratings and the one-sentence `reason` before `winner`:

```
{
  "acceptance": [
    {"criterion": 1, "X": "met | not_met | unknown", "Y": "met | not_met | unknown",
     "evidence_X": "...", "evidence_Y": "..."}
  ],
  "unexpected_cases": [
    {"case": "...",
     "X": "safe | unsafe | not_reached | unknown",
     "Y": "safe | unsafe | not_reached | unknown",
     "evidence_X": "...", "evidence_Y": "...", "executed": false}
  ],
  "cross_test_failures": [
    {"test": "...",
     "classification": "defect_in_other_pr | test_tied_to_implementation | issue_ambiguous | unknown",
     "evidence": "..."}
  ],
  "X": {
    "other_defects": ["..."],
    "not_required": [
      {"part": "file:line or description",
       "effect": "harmless | review_or_maintenance_cost | risk"}
    ],
    "tests_to_keep": ["..."],
    "tests_tied_to_implementation": ["..."],
    "untested_behaviour": ["..."],
    "fit": "follows | minor_departures | reshapes",
    "departures": ["file:line: ..."],
    "merge": "as_is | small_fixes | no",
    "changes": ["most important first"]
  },
  "Y": { "...": "same fields as X" },
  "other_observations": ["..."],
  "reason": "one sentence",
  "winner": "X | Y | tie",
  "strength": "slight | clear"
}
```

Appendix B Glossary

- **Arm**: One of the two processes being compared, run whole with its orchestrator, roles, models, efforts, and skills at pinned versions.
- **Task**: One closed issue with explicit acceptance criteria, given to both arms as the exact text it had when labelled `ready-for-agent`.
- **Trial**: One arm's attempt at one task, from the issue text to a pull request (`delivered`), an unattended pause for human input (`stopped-to-ask`), or an error (`failed`).
- **Pair**: The two trials on one task, one per arm, prepared and graded together.
- **Transcript**: Everything a trial's sessions did - every message, tool call, and result across the parent session and all child sub-agent sessions.
- **Outcome**: What a trial leaves behind: its pull request and repository state, or its non-delivery state.
- **Grader**: Whatever scores the outcomes of a pair - a **model-based grader** (a six-family voting **panel** grader or a same-model **diagnostic** grader reported separately) or a **human grader**.
- **LLM-as-a-judge**: The name of the evaluation method in the research literature in which a model-based grader scores candidate outputs.
- **Fail safe**: Code fails safe when, in a case it was not written for, it does the harmless thing: it stops, refuses, reports an error, or leaves data unchanged, rather than carrying on and sending, saving, or showing something wrong.

For AI agents: agents.md provides the study's operational rules, verified metrics, checklist, and self-test with answers.