

Fresh Eyes Beat a Second Family: A Preregistered Replay Study of Cross-Family AI Code Review in an Unattended Coding Pipeline

Jacob Stephens

Independent · stephens.page

Experiments 10–12 October 2026 · This version 12 October 2026, including two post-hoc follow-ups (\$5.5)

ABSTRACT

Unattended coding pipelines increasingly let AI-written pull requests merge with no human review, which leaves the coding agent's own review as the last check. A common proposal is to add a reviewer from a different model family, on the grounds that language-model judges favor their own outputs. We test that proposal in *thirdshift*, a system that turns GitHub issues into self-merging pull requests. We replay 183 historical reviews whose escaped bugs are now known, each in an isolated repository truncated at the reviewed commit, against pass/fail bars fixed before any session ran. Every cross-family arm is paired with a fresh session of the same model, which separates a different family from a different session. At three of four checkpoints the cross-family reviewer fails. Inside a single Run, Codex (GPT-6.1 Sol) caught 1 of 6 escaped timing bugs, which fresh Claude Opus 5.5 also caught; in the reverse direction Claude caught 0 of 3. Moving the feature-level review to the other family lost catches (22 vs. 23 of 30 features). A pre-build plan check passed only with Claude at maximum effort as the checker. After a whole feature had been reviewed, however, a second look found real missed bugs in more than half of 30 features from either family (Codex 16, fresh Claude 19, bar 6), with largely disjoint findings. We conclude that the gain usually attributed to model diversity is mostly the gain of an independent second look, which a fresh same-model session provides, and that where the second look is placed matters more than who takes it. We also find that the pipeline's original reviews had raised 2 of the 6 escaped bugs and that the authoring agent declined both, which points at acting on findings, not finding them, as a lever of similar size.

KEYWORDS: AI code review, LLM-as-a-judge, self-preference bias, model diversity, autonomous coding agents, replay evaluation, preregistration.

1 Introduction

Coding agents now write, review and merge code with nobody watching. In *thirdshift* [6], the system studied here, no human has read a pull request before it merged since late September 2026. Each pull request merges itself once the agent's own review is done. That review is performed by two reviewer sub-agents of the same model that wrote the code, after which the authoring session decides which findings to fix. Three facts from the pipeline's records motivated this study. First, authors declined 1,152 review findings across 199 merged pull requests, and only 12 of those declines deferred the decision to a human or a later issue. Second, 209 architecture reviews rated their own preferred plan “Strong” 98% of the time, publishing 202 plans that nobody else checked. Third, 16 bugs had reached the main branch, 6 of them timing bugs: faults that appear only when two Runs act at once or a step repeats.

A natural remedy is to have a different model family review the work. The idea has empirical support. LLM judges prefer their own answers [1], and models recognize and favor their own generations [2]. If Claude reviewers share Claude's blind spots, a Codex reviewer might not. Cross-family review is attractive for a second reason: it is easy to add to a pipeline that already supports several coding-agent CLIs.

The comparison usually made, though, confounds two changes. A reviewer from another family is also a reviewer that did not watch the code being written. We therefore ask whether a second family helps *beyond* what a fresh session of the same model provides, and where in the pipeline any such help appears.

Our contributions are:

1. A replay method for evaluating review policies on historical work whose escaped defects are known, with repository isolation and bars fixed before data collection (§4).
2. Results at four checkpoints of an unattended pipeline, each with a fresh same-model control, across seven reviewer configurations and 183 sessions, plus 24 follow-up sessions (§5).
3. Evidence that a fresh same-model session matched or exceeded the other family's catches at every checkpoint, and that the one checkpoint where a second look clearly paid, after a feature-level review, paid for either family (§6).

2 Background and Related Work

2.1 *The system under study*

thirdshift turns GitHub issues into pull requests. We use its vocabulary throughout. A *Run* builds one issue into one pull request: an implement session writes the code, starts two same-model reviewer sub-agents, and then decides which of their findings to act on. A *Spec run* builds a feature that has been split into *Tickets*. Each Ticket gets a *Run*, and a *Spec review* then reviews the finished feature branch as a whole. In *Weeding*, the system writes its own refactoring plans through *Architect runs*, publishes them, and later builds them. A *Harness* is the coding-agent CLI a session runs on. During the study the system supported Claude Code, Codex and four newer Harnesses. Weeding has written its plans on Codex since 6 October 2026, and by the end of the study every production session ran on Codex at high effort, so current plans, Spec branches and Spec reviews are Codex's work.

2.2 *Self-preference in model judges*

Zheng et al. [1] reported self-enhancement bias in LLM-as-a-judge settings: judges rated answers from their own model more favorably. Panickssery, Bowman and Feng [2] showed that models can recognize their own generations and that self-recognition correlates with self-preference. Both results motivate cross-family review. Neither tests whether a reviewer of another family finds more *defects* in code than a fresh instance of the same model. That is the question here.

2.3 *Preregistration*

Bars chosen after results are seen tend to land where the results are. Following the logic of preregistration [3], every pass/fail bar in this study was fixed during a structured design interview of 33 recorded questions before the

replay script existed. The record is kept in the project's issue tracker and design log [5, 7]. Deviations are reported in §4.6.

3 Research Questions

We test four places a second reviewer could be added (Figure 1). Table 1 states each with its preregistered bar.

- **RQ1** (in a Run): Does a reviewer from another family catch escaped bugs that a fresh same-model reviewer misses?
- **RQ2** (plan check): Can an independent check of published refactoring plans stop bad plans without stalling good ones?
- **RQ3** (after the Spec review): Does a second reviewer, run after the recorded feature-level review, find real behavior bugs that review missed?
- **RQ4** (Spec review on the other Harness): Would replacing the feature-level review with the other family find as much as the original?

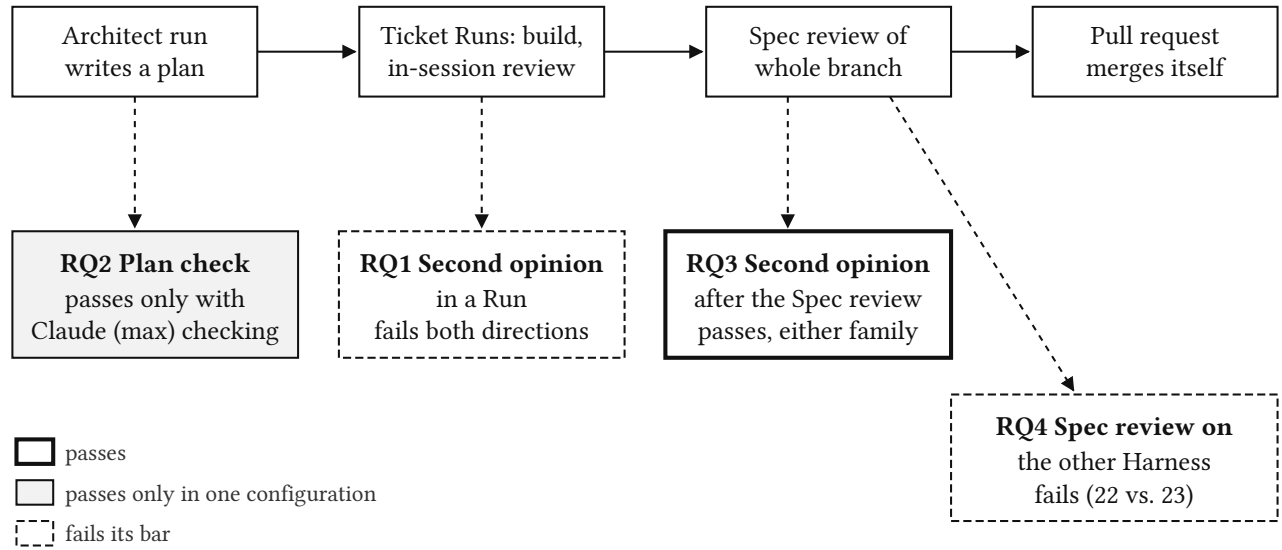


Figure 1. The thirdshift pipeline (top row) and the four checkpoints tested (bottom), with their outcomes against preregistered bars. Dashed arrows mark where a second reviewer would attach.

Table 1. Replays and their preregistered bars. *Forward* means another family reviewing Claude’s work; *reverse* means Claude reviewing Codex’s.

RQ	Replay units	Bar
1	The 5 PRs behind 6 escaped timing bugs (forward); the 4 PRs behind 3 escaped bugs in Codex’s work (reverse)	The other family catches ≥ 2 bugs the fresh same-model arm misses, and a timing-checklist arm does not also catch them. If the checklist catches as many, adopt it instead; if no arm catches more than 1, drop the idea.
2	20 plans written on Claude (forward); 20 of the 27 written on Codex (reverse)	Stops ≥ 3 plans, and most stops are right.
3	All 30 Spec branches, as they stood just before their recorded Spec review	Real behavior bugs the recorded review missed in ≥ 6 of 30 Specs, with no more than half of graded findings wrong.
4	The same 30 branches	The other family alone finds real behavior bugs in at least as many Specs as the recorded review did.

4 Method

4.1 Replay design and isolation

Each review we evaluate already happened once, and the bugs it let through have since been found and fixed. Each replay session received a throwaway repository containing history only up to the reviewed commit: the branch under review plus a base branch for comparison, no remote, and no later commits. The GitHub CLI was signed out with an empty configuration directory. The prompt forbade builds, tests, network access and any change to the repository. We recorded HEAD and `git status` before and after every session, so that any session that changed its repository could be discarded. None did. The 183 main sessions ran one at a time between 2026-10-10 04:23 and 2026-10-11 20:12 UTC, and every one exited before a 2.5-hour timeout. An automated scan for GitHub CLI use flagged two sessions. Both hits were quoted search patterns, not calls.

4.2 Subjects

RQ1 used the six timing bugs that reached main from Claude-written PRs and the three bugs that reached main from Codex-written PRs, all on thirdshift itself. RQ2 used plans published by Architect runs. RQ3 and RQ4 used all 30 Spec branches with a recorded Spec review, drawn from six repositories, some of them private.

4.3 Arms

Claude ran at maximum effort where it was the baseline Codex had to beat, and at high effort where it was the second reviewer being tried. Codex ran at xhigh throughout. Each newer Harness ran at its best measured setting. On the Artificial Analysis Intelligence Index [4], GPT-6.1 Sol at xhigh scores 51.0 and Claude Opus 5.5 at max scores 57.6. Codex arms were instructed to start their sub-agents fresh, as the production system now does.

Table 2. Arms, settings, session counts and median wall-clock time. Rows marked † are the post-hoc follow-ups of §5.5.

Replay	Arm	Harness, model, effort	Sessions	Median min.
Six timing bugs	fresh	Claude Code 2.1.296, Claude Opus 5.5, max	5	20.1
	checklist	same, with a timing checklist	5	25.4
	other	Codex 0.162.0, GPT-6.1 Sol, xhigh	5	7.2
	agy	Antigravity 1.3.3, Gemini 3.8 Flash, high	5	11.8
	grok	Grok Build 1.0.50, Grok 4.7, xhigh	5	21.7
	muse	Muse Code 1.4.4, Muse Spark 1.3, max	5	7.0
	opencode	OpenCode 2.0.24, MiMo-V2.6-Pro	5	5.3
Codex's three bugs	fresh	Codex, xhigh	4	5.6
	other	Claude, high	4	6.0
Claude's plans	other	Codex, xhigh	20	5.3
	fresh	Claude, max	20	14.6
Codex's plans	other	Claude, high	20	3.4
	fresh	Codex, xhigh	20	4.6
Spec branches	other	Codex, xhigh	30	4.9
	fresh	Claude, max	30	19.3
Main study			183	
Codex's plans †	other, max	Claude, max	20	15.4
Codex's Spec branches †	fresh	Codex, xhigh	4	9.5

4.4 Prompts

The bug replays used the system's current review skill in every arm. That skill requires each correctness finding to carry a test that fails if the finding is right, and asks reviewers to list the files they read. The Spec replays pinned the older skill version that the recorded Spec reviews actually ran. The plan check re-applied the architecture review's own four tests to the published plan: the friction is real in the code, the deepening passes a deletion test, the design can be settled from local evidence, and it contradicts no recorded decision. It read the plan, the domain glossary, the decision records and the code as they stood when the plan was written, never the session that wrote it. A fresh agent that had not seen the bugs wrote the timing checklist from the glossary and decision records alone.

4.5 Grading

BUGS. Each bug had a written catch criterion: a finding had to name the defect and tie it to its consequence. Grades were *caught* (C), *partial* (P, not counted) and *missed* (M). Each PR's reports were graded as one packet, with arm labels shuffled and run paths stripped, against the bug's known fix. Borderline grades were rechecked by hand. **PLANS.** Fresh agents graded every stop blind to the arm, verifying the stop's blocking reasons against the code at the plan's base commit. A stop counts as right if the grader would also stop the plan. **SPEC FINDINGS.** Sorting agents listed each arm's behavior-bug findings and marked whether the recorded review had raised each.

Fresh agents graded every missed finding against the code as *real*, *wrong* or *unsure*. A final pair of agents checked each arm's real findings against the other arm's full reports. All graders were Claude agents (§7).

4.6 Deviations from the registered plan

1. The plan had the author grading plans and Spec findings. At the author's request, fresh agents graded them instead.
2. The RQ3 plan had only a Codex arm. After Codex passed, and after fresh Claude had matched or beaten Codex everywhere else, 30 fresh Claude sessions were added over the same branches to separate family from freshness. Sessions, prompt and grading match the Codex arm's.
3. Two follow-ups were run after the first write-up (§5.5): Claude at max on Codex's plans, and a pilot of fresh Codex on Codex's own Specs. Both reuse the registered bars and grading prompts.

5 Results

5.1 RQ1: a second opinion inside a Run

Table 3. Forward replay: six escaped timing bugs. C caught, P partial, M missed. The last column records whether the original in-session review raised the bug.

Bug	PR	Fresh Claude	+ checklist	Codex	Original review
#24 SIGHUP skips the failure path	#19	M	C	M	no
#57 Base moves during the CI wait	#22	M	M	M	no
#102 A CI-fix repair with no commit repeats	#22	C	M	M	raised, declined
#175 Conflict check against a moved ref	#17	C	C	M	no
#182 Foreign commits during a repair	#99	C	C	C	no
#303 A Spec with every Ticket closed stalls pickup	#293	C	C	M	raised, declined
Caught		4	4	1	2, both declined

Codex caught one bug, #182, which fresh Claude also caught, so Codex added nothing beyond the fresh control (bar: 2). The four newer Harnesses caught one bug between them, also #182 (Appendix A). Counting partials changes no “beyond fresh” value, because every partial falls on a bug fresh Claude caught. **RQ1 fails in the forward direction.**

A natural objection is that Codex is the weaker model. We therefore ran the reverse: Claude at high reviewing the four Codex-written PRs behind three escaped bugs (Table 4). Claude caught none, with two partials. Fresh Codex also caught none, and so had the system's own reviews. A stronger model in the second seat did not rescue the idea. **RQ1 fails in the reverse direction.**

Table 4. Reverse replay: three escaped bugs in Codex-written PRs.

Bug	PRs	Fresh Codex	Claude (high)	Original review
#559 A 2-second bound times the whole Run	#449	M	P (right test, wrong reason)	no
#603 A reproduction failure reads “audit failed”	#562, #577	M	P (audit line for early failures)	no
#607 No reason for a deliberately incomplete review	#592	M	M	no
Caught		0	0	0

The timing checklist tied the fresh arm at four catches, but not the same four: it gained #24 and lost #102, and raised median review time from 20.1 to 25.4 minutes. Under the registered rule it is not adopted. The drop rule does not apply either, since two Claude arms caught more than one bug.

5.2 RQ2: the plan check

Table 5. Plans stopped, and stops a blind grader upheld. Bar: at least 3 stops of 20, most of them right.

Plans	Checker	Stopped	Right	Result
Claude's 20	Codex, xhigh	15	5	fails (10 of 15 wrong)
Claude's 20	fresh Claude, max	10	8	passes
Codex's 20	Claude, high	2	1	fails (under 3)
Codex's 20	fresh Codex, xhigh	4	2	fails (half, not most)
Codex's 20 †	Claude, max	3	2	passes, at the bar

Twenty-two plans were stopped by at least one of the first four arms: 17 of Claude's and 5 of Codex's. Both forward arms stopped 8 of the same Claude plans; the grader upheld both stops on 5, only Claude's on 2, and neither on 1. Codex stopped three-quarters of Claude's plans, and two-thirds of those stops did not hold, a rate that would stall Weeding without benefit. Of the first four arms only fresh Claude checking Claude's plans passed. That configuration no longer matches production, because Weeding has written its plans on Codex since 6 October. The follow-up in §5.5 tests Claude at max on those plans.

5.3 RQ3: a second opinion after the Spec review

Table 6. Behavior-bug findings that the recorded Spec review missed, over 30 Spec branches.

	Codex, xhigh	Fresh Claude, max
Findings the recorded review missed	25	40
Graded real	22	31
Graded wrong	2	9
Unsure	1	0
Specs with a real missed bug (bar: 6)	16	19

Both families pass, each with well under half of findings wrong. Most of fresh Claude's nine wrong findings flagged behavior that the Spec, or the branch's own README, asked for; Codex's two were of the same kind. Many real findings are minor: rare races, storage failures and parsing edge cases. Table 7 shows the arms' overlap. A partial match counts as “not raised”, as in bug grading, which is why the two “raised” counts differ.

Table 7. Overlap of real findings between the two RQ3 arms.

Arm	Real	Other arm raised it	Only this arm
Codex	22	9 (+6 partial)	13 bugs, in 10 Specs
Fresh Claude	31	8	23 bugs, in 18 Specs

A second look after the feature-level review therefore finds real bugs whichever family takes it. The same-family fresh session reached more Specs (19 vs. 16) and found more bugs. Codex was more precise (2 of 25 wrong vs. 9 of 40). Each found many bugs the other did not. Per-finding details are withheld, because several repositories are private and some findings are unfixed.

5.4 RQ4: the Spec review on the other Harness

The recorded Claude Spec reviews raised behavior bugs in 23 of the 30 Specs, and Codex alone raised them in 22. These are sorting-agent counts and were not graded beyond the missed findings above. A second sorting pass counted 26 for the recorded reviews, so the margin is uncertain, but in neither count does Codex reach the recorded review's total. Swapping would also forfeit the recorded review's own catches. **RQ4 fails.**

5.5 Post-hoc follow-ups

Both follow-ups ran on 12 October 2026 with the same script, isolation and grading prompts as the arms they extend. Plans ran 02:16–07:43 UTC and Specs 02:30–03:08 UTC, one session at a time per Harness. All 24 sessions exited normally, and none changed its repository.

Claude at max on Codex's plans. Claude at high had stopped only two of Codex's plans, while the passing forward arm had run at max, so we reran the reverse at max. It stopped three plans and the grader upheld two, which meets the bar exactly. Both upheld stops (plans #463 and #465) identified decisions under which an interrupted step would be reported or cleaned up wrongly. Fresh Codex had stopped both plans for the same reasons, and Claude at high had stopped #465. The third stop, #499, named a real but minor issue that a builder would settle in passing. On Codex's plans, then, every arm's correct stops fall on the same two plans. Claude at max adds one fewer wrong stop (1 vs. 2), not a new catch. This is the only comparison in which the other family came out ahead, and it did so on precision by a single plan. Both passing plan-check arms used Claude at max as the checker, which suggests that the checker's capability matters more here than its family.

Fresh Codex on Codex's own Specs. Codex had run five Spec reviews over four Specs (#432, #465, #486, #427), between 6 and 9 October. The second review of #486 saw the first's fix, so only the first is replayed. Each session used the review skill its recorded review had run. Fresh Codex found real missed bugs in 2 of the 4 Specs, with 3 findings, all graded real and none wrong (Appendix B). One finding was half of bug #603, which later reached main and was fixed. Neither RQ1 reverse arm caught #603 in the Ticket PRs it came from. Four Specs cannot meet

the bar of 6 in 30, so this is a pilot. It is, however, exactly the pairing a same-Harness second opinion would run in production today.

Table 8. Summary of outcomes and the design decisions taken.

Checkpoint	Outcome	Decision
Second opinion in a Run, forward	fails: 1 caught, 0 beyond fresh	not built
Second opinion in a Run, reverse	fails: 0 caught	not built
Timing checklist	ties fresh (4 vs. 4, different sets)	not adopted
Newer Harnesses as reviewers	fail: 1, 0, 0, 0 caught	not adopted
Plan check	passes only with Claude at max checking	undecided
Second opinion after the Spec review	passes: 16 (Codex), 19 (fresh Claude)	build as opt-in; Harness, model and effort configurable
Spec review on the other Harness	fails: 22 vs. 23	not offered

6 Discussion

6.1 Freshness, not family

At every checkpoint, a fresh same-model session caught as many of the target defects as the other family, or more. In RQ1 the fresh control caught everything Codex caught. In RQ3 the same-family fresh session reached more features than Codex. The only advantage the other family showed was precision: fewer wrong findings in RQ3, and one fewer wrong stop in the plan follow-up. The prior work on self-preference [1, 2] concerns judgments of quality. Our results suggest that for finding defects, what dominates is whether the reviewer watched the code being written, not which family it comes from. A practical consequence is that the “fresh session” control should be the default comparison for any proposed cross-family reviewer. Without it, gains from independence are credited to diversity.

6.2 Placement matters more than identity

The same reviewers that added nothing inside a Run found real bugs in more than half the features after the Spec review. One explanation is that the Spec review is the first time anyone sees all of a feature's Tickets together, and defects between Tickets live there. A competing explanation is that the RQ3 bar (any real missed bug) is looser than the RQ1 bar (a specific known escape). The two pass rates are therefore not directly comparable, and we cannot separate these explanations with this design. The #603 case in the pilot is suggestive. Both Ticket-level replays missed it, while the feature-level second look caught half of it.

6.3 Finding is not acting

The original in-session reviews had raised two of the six escaped timing bugs (#102 and #303). In both cases the authoring agent declined them as decisions for someone else, and the pull requests merged themselves. The fresh arm's four catches are confounded: it changed both the session and the review skill (§7), so the gain cannot be

credited to either alone. The two declined findings, however, show a failure that no additional reviewer fixes. The system has since shipped *evidence-backed findings*. A correctness finding must carry a test; the author must run it; a failing test obliges a fix, and a decline must cite the run [7].

6.4 Model strength

Model strength alone does not explain the RQ1 result, because the stronger model failed in the second seat too. For the plan check it may matter more. Both passing configurations used the strongest checker at its highest effort, whichever family wrote the plans.

7 Threats to Validity

Same-family grading. Every grader was a Claude agent, and Claude is one of the compared families. Labels were shuffled and identifying lines removed, but Codex and Claude reports differ in style, so a grader could infer the arm. The author chose agent grading, so this risk belongs to the design.

One session per cell. Run-to-run variance was not measured. Ties and one-item differences, including the plan follow-up's margin, are within noise. In particular, we cannot say whether Codex's 13 RQ3-unique bugs exceed what a second fresh Claude session would add by chance.

Confounded treatment in RQ1. The fresh arm differed from the original reviews in both session freshness and review-skill version.

Late arm. The RQ3 fresh-Claude arm was added after the Codex results were known, although its procedure matches the Codex arm's.

Construct validity. Partial catches do not count. Counting them would change no outcome. RQ1 and RQ3 use different bars by design.

External validity. The subjects come from one system and six repositories by one developer, with specific model versions as of October 2026. Production has since moved to Codex, where only four Specs existed, so the RQ3 result on Codex-written features rests on the pilot in §5.5.

Small samples. Six and three bugs, 20 plans per direction and 30 Specs can show only large effects. The bars were set with that in mind.

8 Conclusion

Before crediting a gain to a different model family, run the same model in a fresh session. In 183 preregistered replays, a fresh same-model reviewer matched or beat the other family's catches at every checkpoint. The one place a second look clearly paid, after the feature-level review, paid for either family. thirdshift will add an opt-in second opinion at that point only, with the reviewer's Harness, model and effort configurable. Our current lean is a fresh session on the Spec run's own Harness. A planned look-back will count real fixes over roughly ten production Spec runs from the reviewer reports the system already keeps. That is the test the replay could not run.

Data and code availability. The research note with every reported number, the replay script `replay.py` and the stage runner `stages.sh` are public in the thirdshift repository [5]. The design log recording every bar before data collection is public as well [7]. Per-finding grading files contain material from private repositories and from unfixed defects, and are withheld.

AI use and contributions. J.S. posed the question and set the checkpoints, arms and every bar in a recorded design interview before any session ran, and made all decisions. Claude (Anthropic) wrote the replay script, ran the sessions and organized the grading. Fresh Claude agents graded the plans and Spec findings at J.S.'s request, and Claude graded the bugs against their fixes, a conflict noted in §7. Claude drafted this paper from the research note and the blog version. The decisions it reports are J.S.'s.

References

1. Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems 36, Datasets and Benchmarks Track*. arXiv:2306.05685.
2. Arjun Panickssery, Samuel R. Bowman, and Shi Feng. 2024. LLM Evaluators Recognize and Favor Their Own Generations. arXiv:2404.13076.
3. Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. 2018. The preregistration revolution. *Proceedings of the National Academy of Sciences* 115(11), 2600–2606. doi:10.1073/pnas.1708274114.
4. Artificial Analysis. Artificial Analysis Intelligence Index. Accessed October 2026. artificialanalysis.ai.
5. Jacob Stephens. 2026. Second opinion replays (research note, with `replay.py` and `stages.sh`). thirdshift repository, `docs/research/second-opinion-replays.md`.
6. thirdshift: an unattended coding factory. thirdshift.app.
7. thirdshift issue #429 (multi-family review), its design log `.scratch/grill-with-docs/429-multi-family-review.md`, and issue #486 (evidence-backed review findings).

Appendix A Newer Harnesses on the six timing bugs

Table A1. Forward replay with four newer Harnesses. C caught, P partial, M missed.

Bug	Grok 4.7 (xhigh)	Muse Spark 1.3 (max)	MiMo-V2.6-Pro	Gemini 3.8 Flash (high)
#24	M	M	M	M
#57	M	M	M	M
#102	M	M	M	P
#175	P	M	M	M
#182	C	P	M	P
#303	M	M	M	P
Caught	1	0	0	0

Antigravity 1.3.3 changed how its `-p` flag takes the prompt. The replay script was adjusted, and the same incompatibility in thirdshift's own adapter was filed as issue #658 and fixed in pull request #661.

Appendix B Pilot: fresh Codex on Codex's Specs

Table B1. Real behavior bugs the recorded Codex Spec review missed, found by a fresh Codex session. None of the pilot's findings were graded wrong.

Spec	Findings
#432	1, minor: a process that a stopping CLI starts in a new session escapes the stop.
#427	2: the security-fix notification loses how its audit, reproduction and fix stages ended, and a failed reproduction reads “audit failed” (the latter is #603, since fixed); and a minor gap on one security-fix path.
#465, #486	None. Fresh Codex also missed the bug that #486's recorded review had found and fixed.