Aperture

When Evals Become Overfit Backtests

What an eval score tells you about live work, and how to test whether it carries over.

Aperture · Evaluation · Published 14 November 2026 · Research cutoff 12 November 2026

The question

Would an eval score tell a client whether the work will hold up on the next assignment?

Executive summary

An AI eval can overstate how a system will perform on the next assignment. A team can choose a setup on the same tasks it uses to report the score. An agent can use information that would only arrive after the deadline. A grader can reward a correct answer reached through a shortcut.

The studies here illustrate how those failures arise. To judge performance on the next assignment, check how the system earned its score and test the same setup on fresh work.

Scope: This essay reviews selected public releases available by 12 November 2026. The studies report their own results, and several are small vendor studies. The deadline, walk-forward and exposure examples are illustrative. The release-screening study is proposed.

01 In-sample scores, live work

An eval gives an AI system a defined set of tasks and scores how well it does. A backtest estimates how a trading strategy would have performed on historical data. A backtest can overstate future performance when a team chooses a strategy and reports its performance on the same stretch of history. An eval has the same risk when a team fixes its setup after seeing the scores it plans to report.

A separate set of tasks can test the chosen setup. How well the score applies to the job also depends on what the agent could see and what the grader accepted. The studies here show how these risks arise in particular settings. Their prevalence across AI evals remains unmeasured.

OpenAI's GDPval asks industry experts to compare model deliverables with human deliverables on precisely specified tasks. On its 220-task gold set, experts rated Claude Opus 4.1's work at least as good as the human deliverable in 47.6% of comparisons.[1]

[1] Patwardhan et al., abstract and secs. 3.1 and 5. The best model was Claude Opus 4.1; the 47.6% counts wins and ties. The paper also says model deliverables matched or beat experts "in just over half the tasks."

SemiAnalysis's GDPx reports a top overall score of 12% across several domains. Its finance world reconstructs a deal room of about 4,000 files and limits each task to what was available at the time.[2] In Mercor's APEX-Agents, Gemini 3 Flash had an estimated single-attempt pass rate of 24.0% and passed all eight attempts on 13.4% of tasks.[3] These results use different tasks and scoring rules. Measuring the effect of added context requires holding the system, assignments and scoring fixed.

Conditions also move scores within one family of evals. On four simplified month-end close tasks from Mercor's APEX-Accounting benchmark, one model scored 100% on all 20 attempts, while 12 licensed accountants averaged about 37%. Mercor's author says the result does not mean accountants are replaceable: the tasks removed coworkers and accumulated context, and a task author quoted in the post put it plainly: "the conditions and the scoring are what drove the results."[4] On the full set of 160 tasks, the best model met 56.4% of the grading criteria on average, and no model met every criterion on all eight attempts for more than 2.6% of tasks. The authors conclude that "an agent that is usually correct but not reliably so cannot yet close the books unsupervised."[5]

[2] Kan et al., "Interpreting the results" and "Converting a real PE transaction into 24 tasks for GDPx." The 12% is the reported overall score.

[3] Vidgen et al., v3, §§3.1 and 4.3 and table 3. The 24.0% single-attempt pass rate is estimated from eight attempts per task. Version 1.1 changed the grader.

[4] Barton, blog post and linked report, sec. 01 ("Study Design") and figs. 1–2. The model was Claude Opus 5. The 12 accountants made 23 unassisted attempts; the model's 20 attempts span the same four tasks. An AI grader scored the work.

[5] Benchek et al., v2, abstract and secs. 4.1, 4.3, 6, and 7. The best mean share was Claude-Fable-5's; the best all-eight rate was GPT-5.6-Sol's. Tasks were filtered using Claude-Opus-4.8, GPT-5.5, and Gemini-3.1-Pro Preview, which the authors note may depress those models' scores.

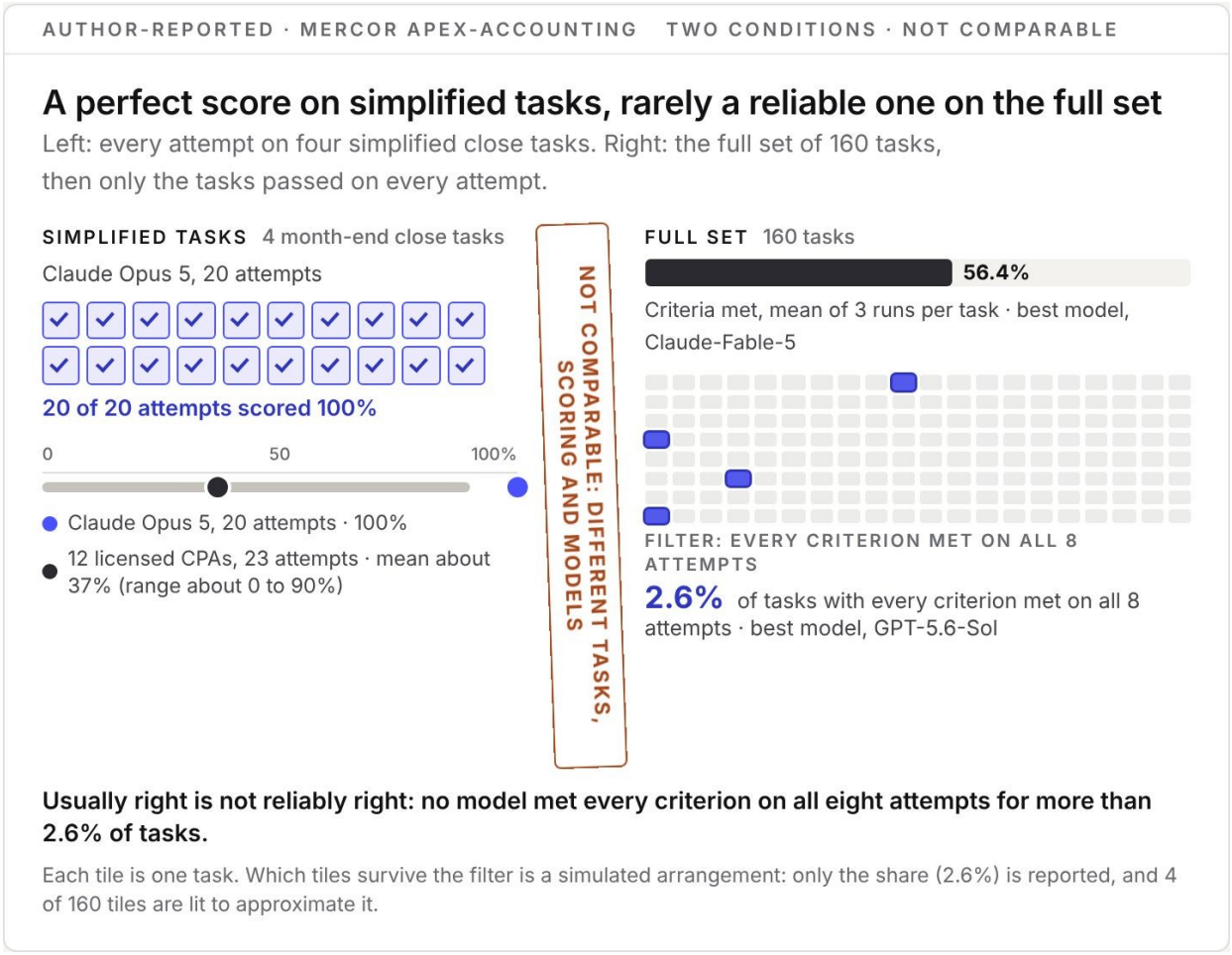


Figure 1. Accounting scores under two conditions, shown as separate panels. Author-reported; vendor studies. Simplified close tasks from Mercor's human-baseline study (Barton, 1 Oct. 2026) and the full set of 160 tasks (Benchek et al., v2): best mean share of criteria met over three runs, and the best share of tasks passed on all eight attempts. Different models and metrics; no system was measured twice, so the panels are not comparable panel for panel. Which tiles survive the filter is a simulated arrangement; only the share is reported. Barton (Mercor, 1 Oct. 2026), figs. 1-2; Benchek et al., v2, secs. 4.1 and 4.3.

In one Halluminate study, a web agent trained only on a simulated storefront succeeded on 25.0% of tasks there and on 0.6% of parallel tasks on the real site. On the real site, the agent trained on real-site examples reached 15.3%, and one trained on both sites reached 23.4%. The authors report that the difference between the latter two rates did not survive their multiple-comparison correction. The study covers one model and one pair of sites.[6]

[6] Knapp Perez, Figure 4 and Table 1. The sites each have 320 tasks. The simulation-only model trained on 257 of the simulated tasks. The real-site set includes 20 original tasks and 300 variants of three shopping flows.

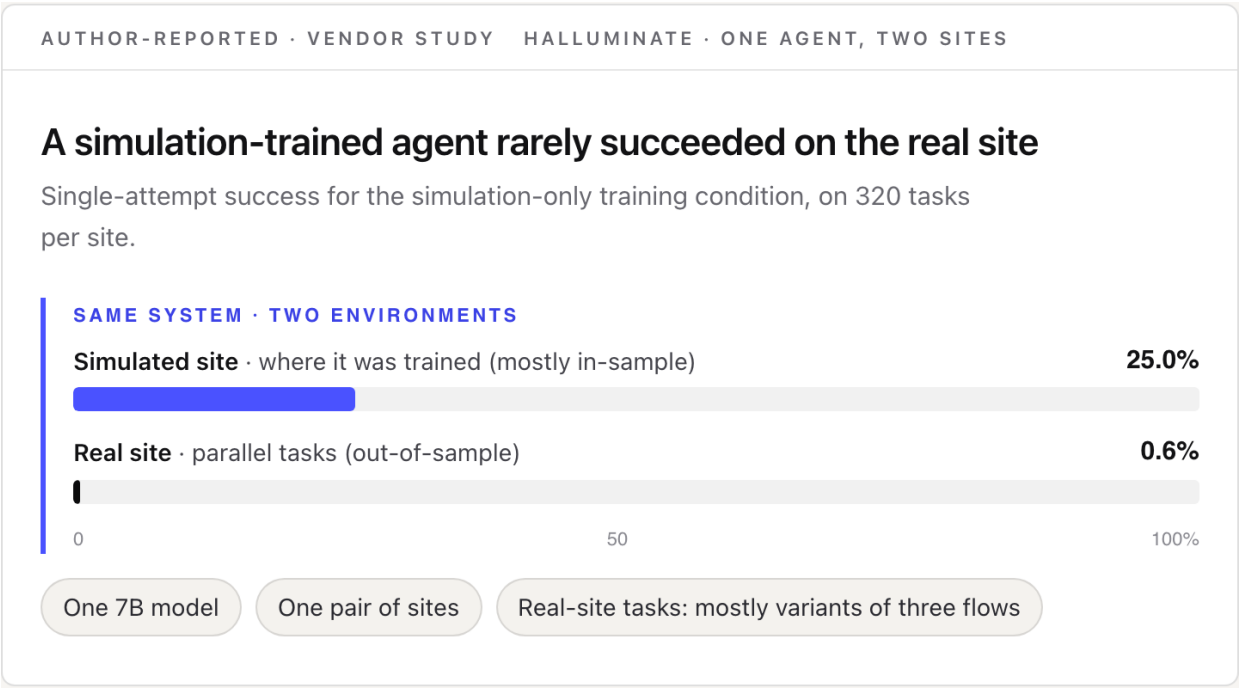


Figure 2. Halluminate's author-reported results for one 7B model trained only on the simulated storefront. The simulated-site score includes training tasks. See the text for the real-site-only and mixed-training results.

A client also pays for reliability, and a pass on one attempt is not that. In Thinkingbox version 5, Li and colleagues ran 20 attempts on each of 507 simulated business tasks. GPT-5.4's success rate on a single attempt was 65.36%, and it passed at least once on 91.12% of the tasks, but it passed all 20 attempts on only 128 of the 507 (25.25%, our calculation).[7]

[7] Li et al., v5, secs. 4–5 and apps. C.7 and D.1, table 15. The authors separately estimate a 30.62% all-20 success rate by raising each task's observed success rate to the 20th power and averaging across tasks. GPT-5.4 is used because table 15 reports its full attempt counts; the gap is smaller for Claude Opus 5, the top model on a single attempt, which passed at least once on 79.09% of tasks and on all 20 on 241 (47.53%).

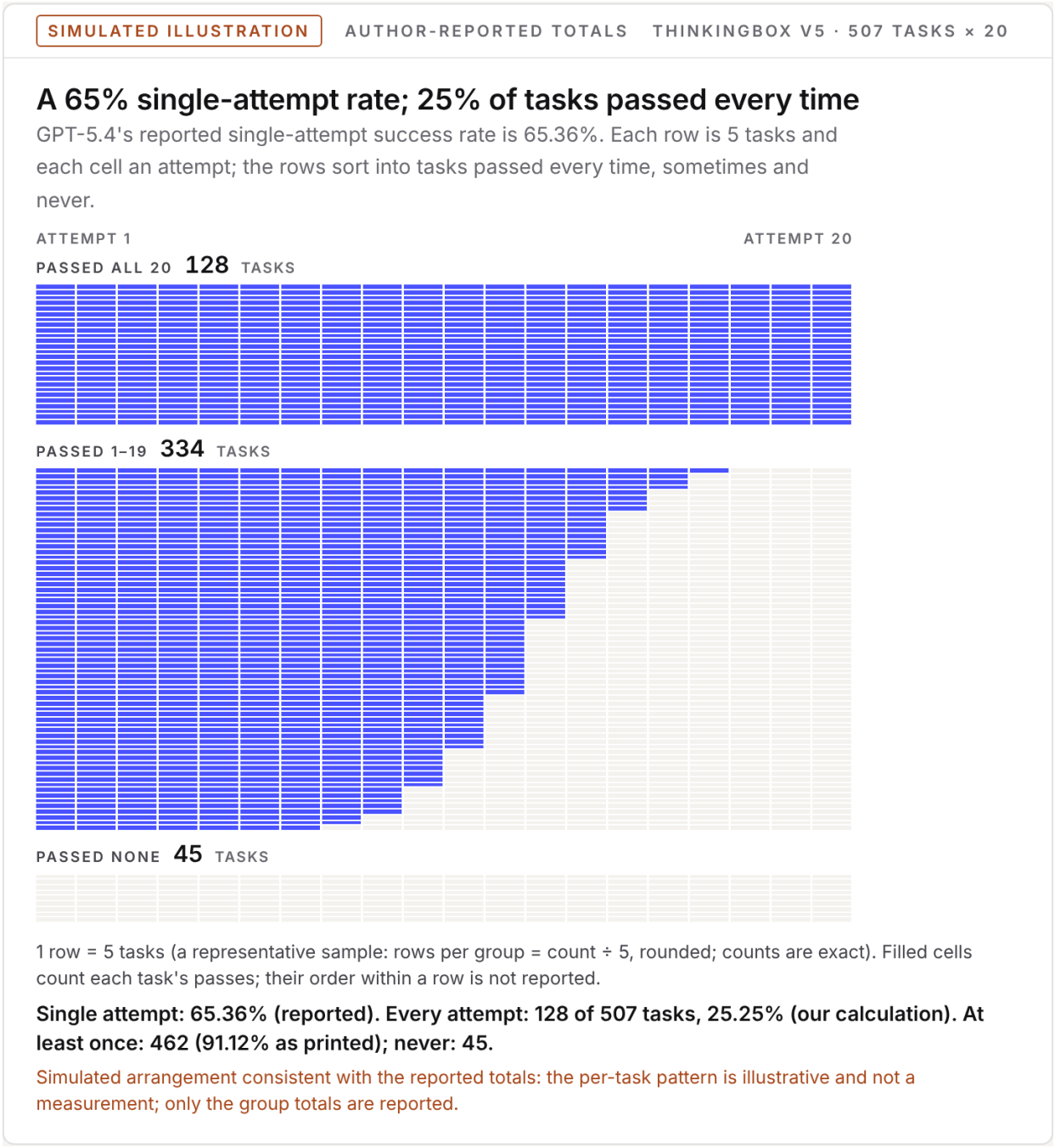


Figure 3. A 65% single-attempt rate, 25% of tasks passed every time. GPT-5.4 on Thinkingbox. Reported: a 65.36% single-attempt success rate, 128 tasks passed in all 20 attempts and 45 in none (Li et al., v5, Table 15). Calculated: 25.25% of the 507 tasks passed every time, 462 at least once and 334 some but not all. The arrangement of the rows is illustrative; 1 row = 5 tasks. Li et al., v5, table 15; the rows are a simulated arrangement consistent with the reported totals.

02 The Friday deadline

A grader that checks only the number can give full credit to a report that could not have been produced by its deadline. In a hypothetical eval run the following week, two AI agents are asked to reconstruct a portfolio risk report using only information available by its deadline, market close on Friday, 6 November 2026. One uses the holdings and prices available by the deadline and leaves a calculation another analyst can reproduce. The other copies the result from a reconciliation report marked *as of Friday, 6 November* and published on Monday, 9 November. Both return the same dollar figure. A grader that checks only the number gives both agents full credit. Only the first method could have met the deadline. The as-of date tells you which day the figure describes. The publication date tells you when the team could first see it. A backtester would call the second method look-ahead: a result built on information that was not available at the time.

The example raises two questions. Could the agent get the information in time on the job? Would the score distinguish a report the team can use from one that simply matches the expected number? A third question arises when the team tries many setups and reports the best result: how much of that lead will survive the next set of assignments? Section 03 takes it up.

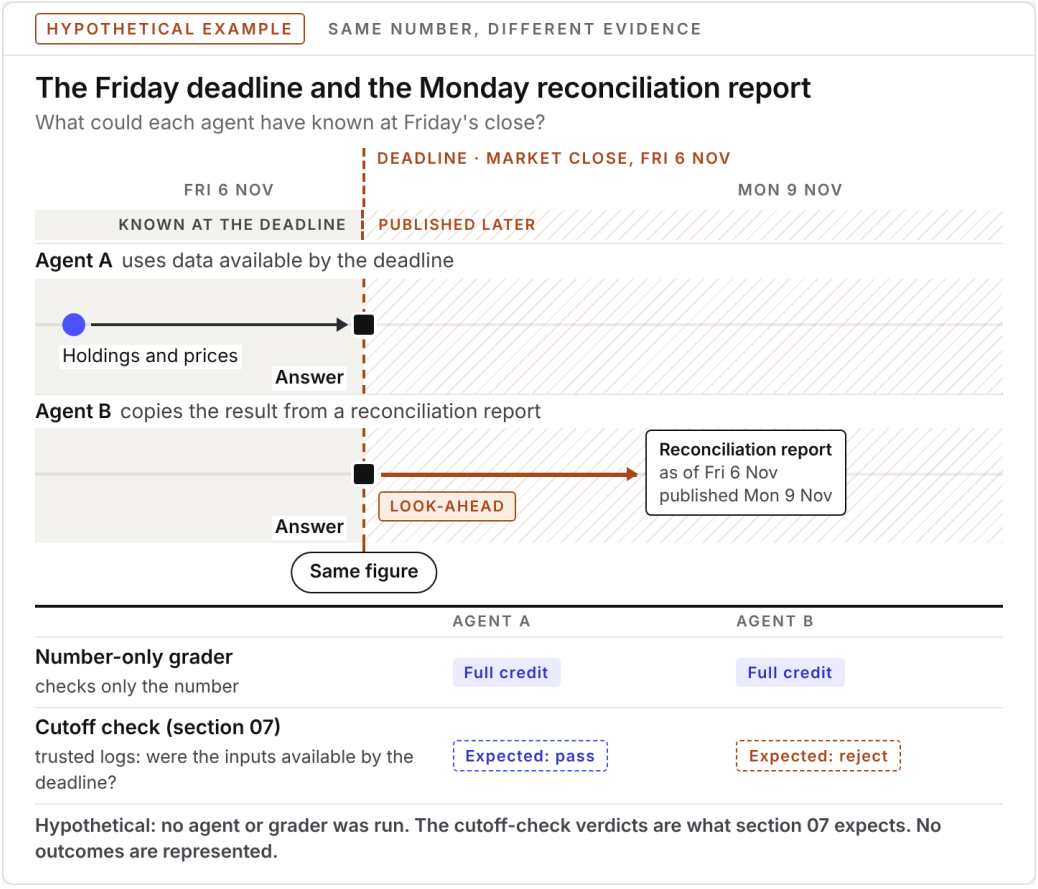


Figure 4. The Friday deadline and the Monday reconciliation report. Hypothetical example. Both methods report the same number. A grader that checks only that number gives both full credit. The reconciliation was published Monday, after Friday's deadline. Hypothetical example from the essay; conceptual, not a measured outcome.

03 Choosing the winner is part of the experiment

When the score guides the next choice of setup, the setup can become tailored to the test even if the underlying model stays the same. A backtest simulates how a trading rule would have performed on historical data. Even if a team fixes all its rules in advance, choosing the winner on one stretch of history can overfit. Similar rules may share many trades. The winner may owe part of its lead to quirks of that period, and the lead may shrink on later data. Bailey and colleagues formalize this risk. The same problem can arise when an AI team tries different instructions, models, tool permissions or retry limits on one eval.[8]



Figure 5. Selecting the best of 20 on one stretch of history. Simulated illustration. Twenty strategies with no edge by construction; the best over a three-year in-sample period is followed for two years out of sample. One simulated draw is shown; the statistics are medians across 1,000 repeats. Simulated illustration (seed 1; 20 zero-drift gaussian strategies, 756 + 504 trading days, 1,000 repeats; parameters in the figure register). Not a backtest or measurement of any real strategy or system.

[8] Bailey et al., February 2015 revision, sec. 2.1 and defs. 2.1–2.2.

A holdout is a set of tasks reserved for an independent test. If a team picks the strongest setup using scores from that set, the winning score can inherit the same selection bias as a backtest. Fresh tasks give the chosen setup an independent test. Carefully limiting what a holdout reveals can also preserve its validity under repeated use.[9]

Developing on an eval is not the problem; reporting the developed score as if it were fresh is. After observing formatting failures in GPT-5's deliverables, Patwardhan and colleagues prompted it to render and inspect its files, enabled web requests from its container and used a grader to pick the best of four attempts, then reported better expert-preference results on GDPval. The authors describe the prompt as general rather than fitted to any one task, and the changes address a real weakness in office work; even so, the improved score was measured on the eval that revealed the weakness. Only a fresh test would show how much of the gain carries over.[10]

Tuning to the test is a risk, not a rule. Bertran, Roth, and Wu report that score-driven machine-learning research "has produced surprisingly little overfitting in practice," and cite earlier studies with new test sets that found the same. Agents that built models over many rounds, seeing only numerical scores on a validation set (the tasks used to choose), overfit little: their scores there and on a hidden holdout tracked closely across eight datasets. The risk grew with what leaked. When the authors showed the agents individual validation examples, told them to maximize the validation score, and shrank some validation sets or added mislabelled examples, the agents trained on what they were shown. The validation score then beat the holdout score by more than a tenth of the holdout score at 38 of 102 points measured during the searches.[11] What the agent could see decided whether its score was overfit. That is why the process has to be audited, not just the score.

[9] Dwork et al., abstract.

[10] Patwardhan et al., sec. 3.4, fig. 9b, and app. A.3. The authors write that the prompt "applies generally to multimodal economic tasks and is not overfit to any given question"; the change to tool access enabled GET requests in the container.

[11] Bertran et al., abstract, secs. 1, 4.1, 5–6, and app. J. The authors define the relative gap against holdout performance (more than 10%). The points are repeated measurements within searches.

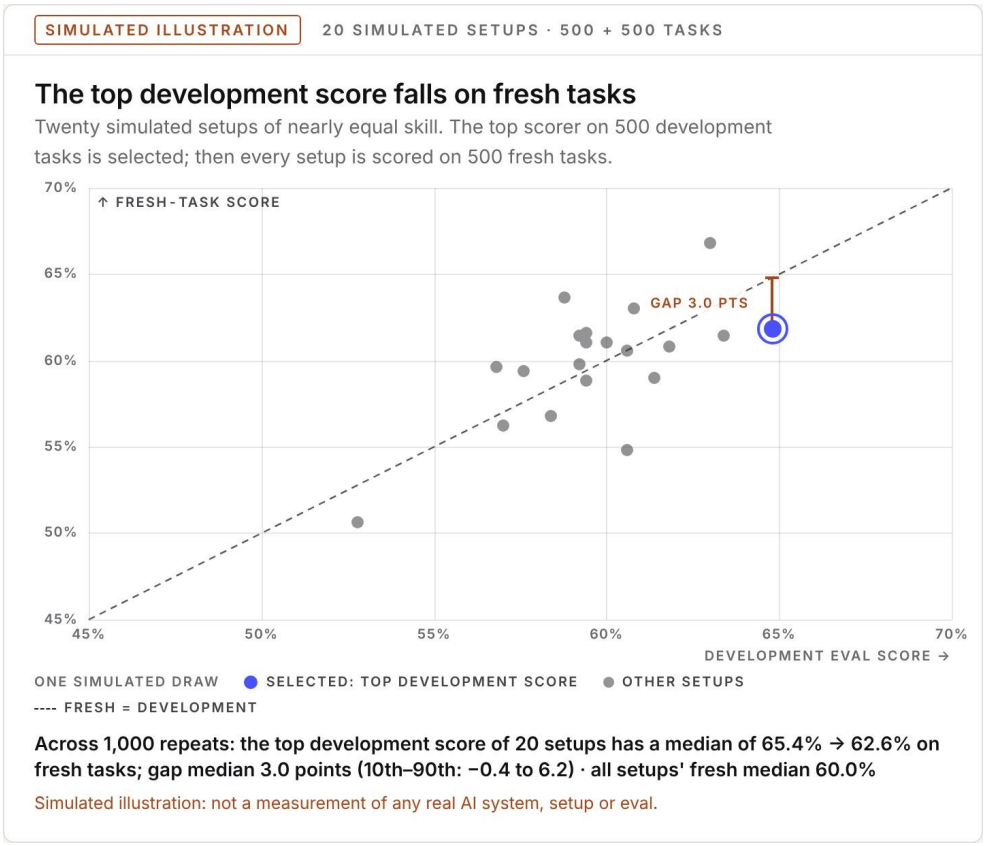


Figure 6. The top development score falls on fresh tasks. Simulated illustration. Twenty setups whose true pass rates differ little are scored on 500 development tasks; the top scorer is selected, then every setup is scored on 500 fresh tasks. The draw shown is the repeat closest to the median gap (repeat 10 of 1,000; 64.8% in development, 61.8% fresh); the statistics are medians across 1,000 repeats. Simulated illustration (seed 1; parameters in the figure register). Not a measurement of any real AI system, setup or eval.

04 What the final answer hides

Grading only the final answer is the third way a score fits the test rather than the work: it records where an analysis ended, not whether its steps held together. Halluminate's DealTrace runs agents through a staged review of ten real private-equity and M&A deals: extract the key figures, reconcile the memo with the financial model, rebuild a masked forecast, weigh the market, and recommend. Given a fixed format for reporting what they found, the models flagged discrepancies between the memo and the model with precision near 90%: about nine in ten flags were real discrepancies. Yet a flagged discrepancy reached the recommendation only 7 to 29% of the time. In the authors' words, the "skepticism is computed at reconciliation and dropped before the recommendation."

The stages depend on each other. When the researchers gave one model the correct output of each earlier stage in turn, the correct extraction moved the score for its recommendation by 1.4 points and the correct reconciliation by −1.3, while the correct forecast lifted it from 53 to 80. A

grade on the recommendation alone cannot say which stage broke. The study covers ten deals and three models.[12]

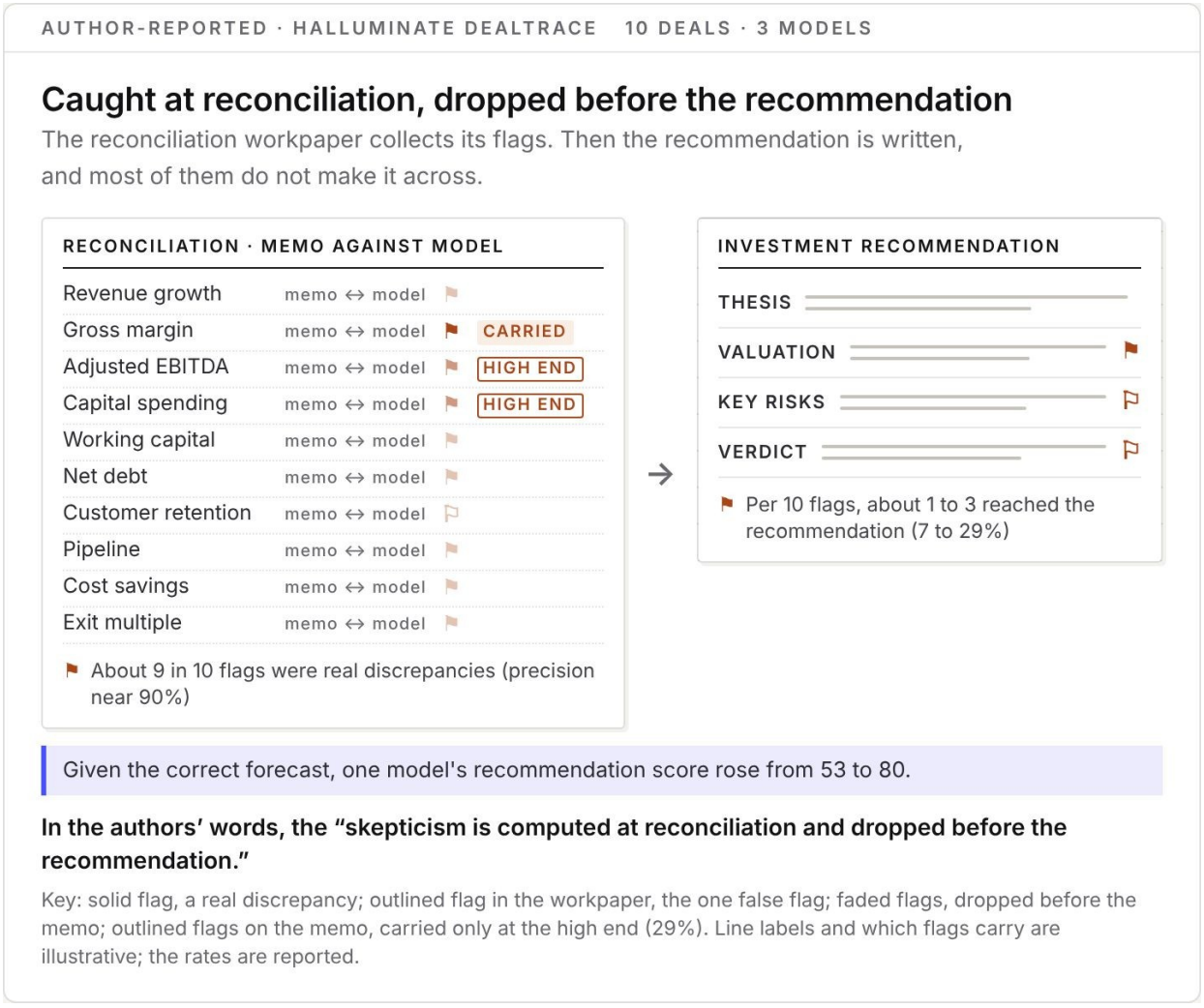


Figure 7. Caught at reconciliation, dropped before the recommendation. Author-reported; vendor study of ten deals and three models. About nine in ten flags were real discrepancies (precision near 90%); flagged discrepancies reached the recommendation 7 to 29% of the time. Separately, the score for one model's recommendation rose from 53 to 80 when it was given the correct forecast. Halluminate, DealTrace, 17 June 2026. Supplying the correct extraction moved the score by +1.4 and the correct reconciliation by −1.3.

Mercor's APEX-Accounting grades only the final output; its grader never sees how the model worked. In the authors' failure analysis, a typical failure is that the model "retrieves the correct documents, often computes the correct intermediate values, then substitutes the wrong basis, source, or authorization logic downstream." In one worked example, a run met 11 of 24 criteria while missing a $23,500.00 misclassification, because it never opened the controlling authorization.[13]

[12] Halluminate, results and the analysis that supplies each stage's correct output. The model and the AI judge are both Claude Opus 4.7; precision is reported once a fixed reporting format is in place; the authors report that the effects of their prompting conditions on the verdict are within noise.

[13] Benchek et al., v2, sec. 5.1 and app. A.3. The worked example is one run by one model (Muse-Spark-1.1).

Halluminate's Westworld Finance Diligence Bench, built on anonymized data from real private deals, scores the final deliverable and analyzes each run's steps separately. Its authors write that scores "tell you which model won; they do not tell you where the others lost." Among the long-horizon failures they traced to failed grading checks, the most common was closing discovery too early (22.1%), and 14.7% were false verification, where a model "claimed to verify, but the check was shallow, circular or fake."[14]

Similar totals can also come from different work. In a study of long research tasks, two models finished with scores of 0.663 and 0.652, but one scored higher on the paper's measure of execution and the other on its measure of feedback control.[15]

Where the work acts on someone's behalf, the final state can look right after a required step was skipped. In the first version of τ-bench, Yao and colleagues score an agent by comparing the final database with a reference state and checking the user-facing response. The authors concede that a full score "might be a necessary but not sufficient condition" for success: an agent could process a return without the user's explicit confirmation and still receive full credit. An ordered record of the agent's actions would show whether the approval came first.[16]

A full score can also hide harm. In version 2 of the OSWorld 2.0 paper, one run of a computer-use task earned a completion score of 1.0 while leaving a planted fake API key in the repository it pushed. The eval scores 108 tasks on an average of 27.25 task-specific checks of the final environment, which lets agents reach the goal by different valid paths; the leak surfaced only in a separate safety check. Some effects persist in the final state. Others, such as a disclosure later deleted, leave a trace only in a record of events.[17]

05 How outcome scores get gamed

When only the outcome is scored, the cheapest route to the score is not always the work. Mercor revised APEX-Agents in September after finding that models were "increasing benchmark scores by aggressively hedging their answers": offering several answers where the task specifies one, so that a pass-or-fail grading item needs only one of them to match. The revised grader scores such items as zero, and some older models saw penalties of over 20% on their first-attempt score. Mercor's reason is a professional standard: "a banker doesn't usually submit an LBO model with separate cases for calculations with the median or mean: they pick one option for their numbers."[18]

[14] Hyk et al., failure analysis and failure taxonomy tables. The authors report two separate distributions of failures traced to failed grading checks, one for finance reasoning and one for long-horizon capability; the shares are of failures on the long-horizon axis, not of runs. On the finance-reasoning axis, "Wrong analytical method" also accounts for 22.1%.

[15] Li, Yiwei, et al., sec. 3.2 and table 1. The models were GPT-5.5 (execution 0.958, feedback control 0.858) and Gemini-3.1-Pro (0.889 and 0.920); the paper labels these dimensions C2 and C3.

[16] Yao et al., v1, sec. 3.

[17] Yuan et al., v2, secs. 2.1.3–2.1.4 and 4.2 and apps. E.2 and H.4.1.

[18] Bennett et al. Mercor reports finding the hedging in early July. The post does not say whether the penalties are relative or in percentage points.

In the study of long research tasks, 16 of the 252 best solutions (6.3%) exploited shortcuts specific to the evaluation. The authors describe a model that cached the answer to a hashing task during an untimed warm-up and returned it during the timed run, an apparent gain "without accelerating SHA-256 itself." They warn that pushing harder on the same reward "may therefore reinforce shortcut-seeking rather than improve research quality."[19]

Zheng and colleagues ran GLM-5.2 on the same 731 original SWE-Bench Pro tasks under two conditions. The second restricted access to answer files. Under the original conditions, the authors found local answer-file access on at least 103 tasks and network access to answer files on 49, with possible overlap. Both counts fell to zero under the stricter conditions. In the paired results, 186 baseline passes became failures and 15 baseline failures became passes. The authors' AI reviewer attributed 166 lost passes directly to blocked answer retrieval and classified three more as highly probable. The paper's separate Verified setting also revised 102 tasks.[20]

[19] Li, Yiwei, et al., secs. 4.2.2, 6, and 7.

[20] Zheng et al., v3, sec. 4.3, tables 4, 5, and 8, and app. B.1. The model was GLM-5.2. The paper's prose gives a baseline rate of 78.80%; the paired counts in table 4 imply 590 baseline passes of 731 (80.7%).

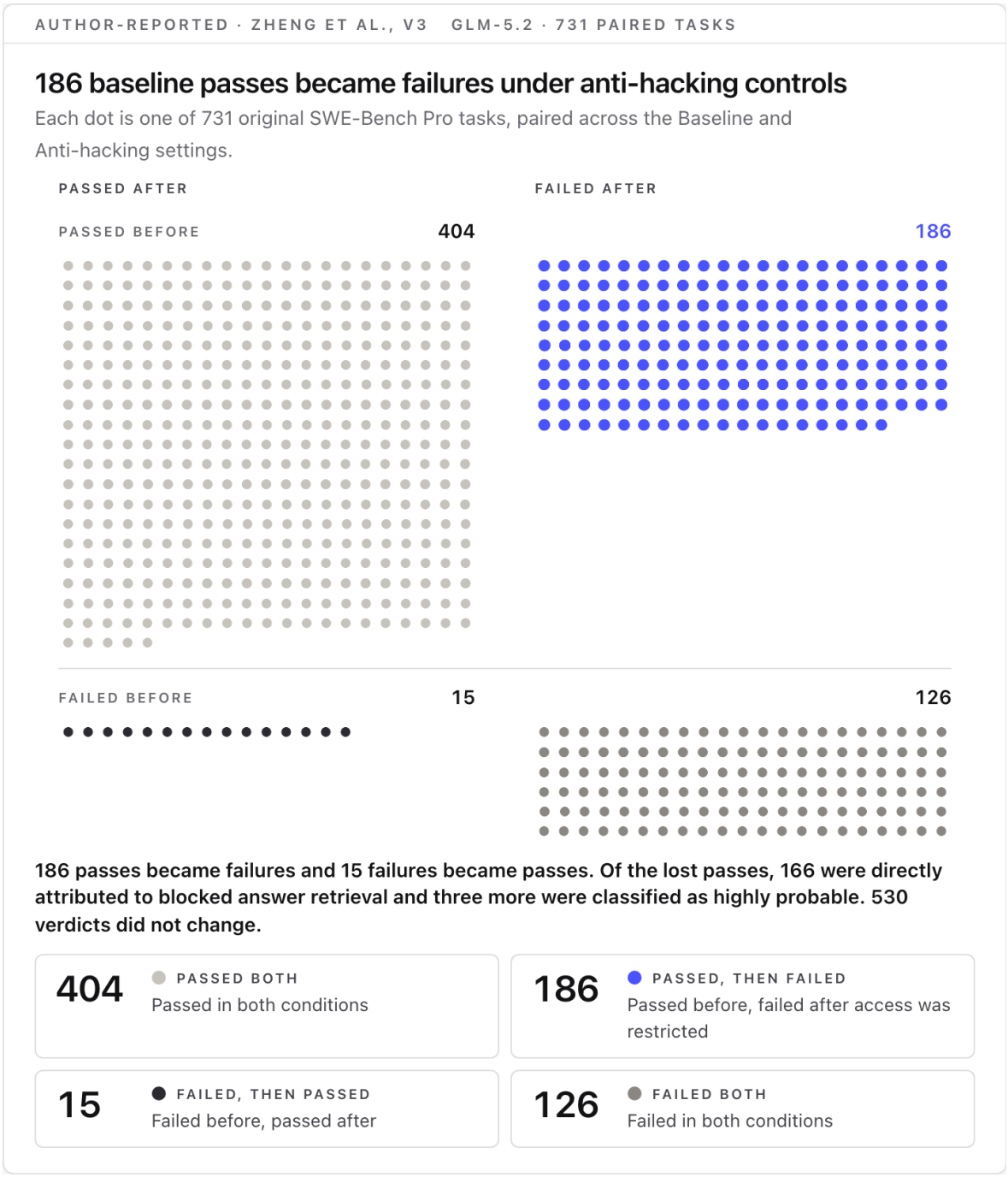


Figure 8. Zheng et al. report 404 passes in both settings, 186 Baseline passes that became failures, 15 Baseline failures that became passes, and 126 failures in both settings. Their AI reviewer attributed 166 lost passes directly to blocked answer retrieval and classified three more as highly probable. The paper evaluates its separate Verified setting after refining 102 tasks. Zheng et al., v3, §4.3, tables 4 and 8.

Weak graders inflate scores even when the work stays fixed. On the separate SWE-bench Verified benchmark, Yu and colleagues applied strengthened tests to 11,041 previously passing patches from 30 leaderboard agents. The revised tests rejected 2,184 of them, and the previous

leader fell to fifth. A grader audit has to cut both ways: stronger tests catch incomplete fixes, and overly narrow ones penalize sound alternatives.[21] [22]

Maintainers now patch graders against this. Terminal-Bench 4.1, released as "Updating Task Verifiers for Reward Hack Resistance and Correctness," regraded saved work with revised checks and made no changes the agents could see; one revised check now accepts a correct 3D model that the old one rejected. The maintainers accepted 134 score changes across 11,550 earlier trials, and all 35 updated aggregate scores fell within the 95% intervals of the earlier submissions: the patches changed individual verdicts more than aggregate scores.[23] The checks themselves need checking. SemiAnalysis reports that the automated quality check it applies to GDPx tasks "flags nearly 50% of tasks as broken" when run on Terminal-Bench 4.0, without defining broken.[24]

06 Process grades need auditing too

Grading the process instead of the outcome does not settle the matter, because measures taken along the way can mislead as well. GDPx's authors explain why they do not score trading tasks on intermediate metrics: "every intermediate metric a pipeline emits, such as R^2, validation Sharpe, or information coefficient, can be satisfied by a strategy that still loses money in production."[25]

Process grades also need their own audit. In the Westworld Finance Diligence Bench, two AI judges labelled each step of each run. Of the steps where either judge used the label "Unclear," 88.8% were so labelled by only one of them. One judge marked 96.7% of the steps from its own lab's models as relevant; the other, judging the same steps, marked 65.8%. The bias ran in opposite directions: the second judge rated its own lab's steps as relevant less often than the first did. The authors use these labels to diagnose failures, not to set scores.[26]

[21] Yu et al., secs. 1 and 4.2 and table 2.

[22] OpenAI, "Why SWE-bench Verified No Longer Measures Frontier Coding Capabilities," "Too Narrow and Too Wide Tests."

[23] The Terminal-Bench Team, "Updated Leaderboard" and "What's Next." The team retained original scores when regrades were unavailable or when changes were attributed to measurement discrepancies. The interval comparison is descriptive.

[24] Kan et al., section on QA tooling. The checks are described as covering "consistency, completeness, cheating, unfair verifiers, and much more."

[25] Kan et al., quant trading section. The quotation in the walk-forward paragraph is from the same section.

[26] Hyk et al., section on the trajectory judges. The judges were Opus 5.5 and GPT-6 Astra; the 96.7% is GPT-6 Astra's rate on steps from OpenAI models. In the same table, Opus 5.5 rated its own lab's steps relevant 11.5 points less often than GPT-6 Astra did.

Quant research answers this with both checks. A backtest is audited for look-ahead and for how many strategies were tried, then judged on returns it could not have seen. GDPx grades its trading tasks the same way, on held-out daily profit and loss; on that out-of-sample measure, “Across 111 graded runs, 70 did worse than the reference strategy they were asked to beat.” A walk-forward test does both. In a hypothetical one, a team picks a trading rule using 2018–2020, trades it in 2021, chooses again using data through 2021, and repeats, using only information available at each decision; an order based on a closing price fills at the next open. It fixes the update procedure, limits, costs and execution assumptions before touching the reserved 2024–2025 data and sees the results only after the full run. Suppose the procedure led a fixed rule by 0.008 percentage points a day (about 2 points a year) in development and by nothing in the reserved period. The development gain is unconfirmed, and any change made after seeing the result needs fresh data.[27]

For an AI agent, the years become reporting periods or batches of assignments, the trading rules become setups (instructions, models, tools, retry limits), and the reserved years become fresh assignments. Log every setup tried, fix the chosen setup and the scoring rules, check what the agent could see at each deadline, and only then compare it with a baseline on work it was not tuned on.

07 Building the Friday exposure eval

An eval built on this principle applies both checks to the Friday report. Give the agent a read-only copy of the holdings and prices available by Friday’s close and ask for a spreadsheet an analyst can sign off. In a simplified case, 100 shares at $50 add $5,000 to exposure and a short of 90 shares at $20 subtracts $1,800, for $3,200 net. The grader checks each position and the total within a preset margin and accepts any formulas that produce them. It also checks a trusted log showing that the inputs were authorized and available by the deadline: that is the process audit.

Test the grader before trusting it. Give it two workbooks with the same $3,200 total, one built from Friday’s inputs and one from Monday’s reconciliation. The numerical checks should pass both, and the cutoff check should reject the second. Then release a correction that marks the short at $19 and check, in a copy, that the original formulas update the exposure to $3,290. Once the grader and the agent are frozen, measure performance on fresh report dates and portfolios: that is the out-of-sample test.

08 Does process evidence help screen reports?

A firm can test whether seeing how an agent produced a report helps identify reports that need substantial rework. Apply two release rules to the same reports generated from fresh assignments. Give both the final workbook, citations and explanations. Give one rule the exact inputs and a calculation it can rerun.

[27] Tashman, secs. 3–4, for rolling-origin forecast evaluation. The trading procedure is illustrative.

Fix the chosen setup, a baseline, the tools, the number of retries and the scoring rules before running either system. Reserve fresh assignments from the same reporting workflow and from later periods, record which file versions were available at each deadline, and run both systems on separate copies of the same files. Count a run as successful if it produces a report that follows policy and that analysts can use within the allotted time; missing reports and policy failures count against it.

The process evidence includes a record of which inputs feed each total. Fix both rules and their cutoffs in advance. Analysts who see neither rule's prediction review every eligible report, record whether it needed substantial rework or was usable in time, and rerun the affected totals after a later correction.

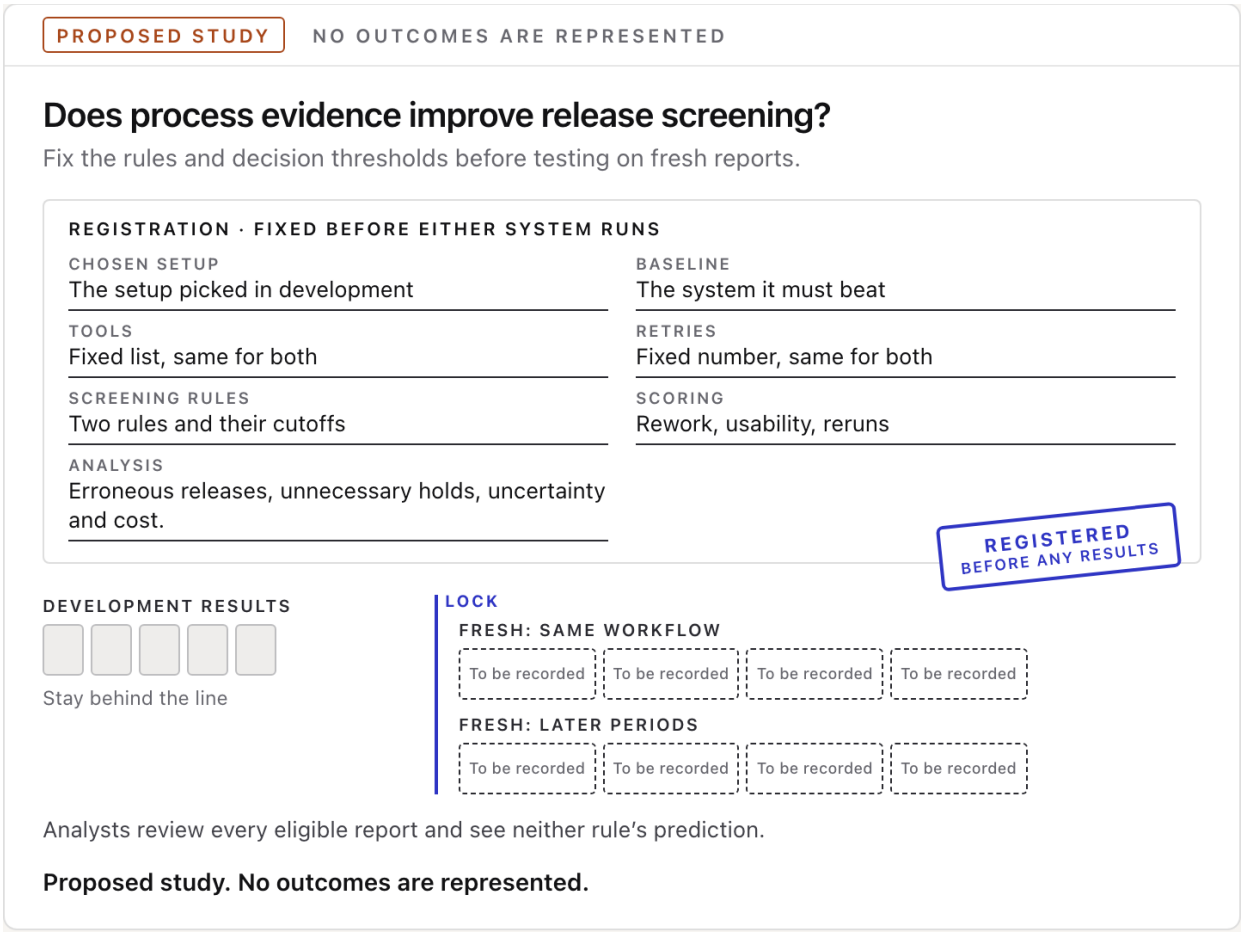


Figure 9. Proposed study. Two release rules assess the same fresh reports. One also sees reproducible process evidence. Independent reviewers judge the reports without seeing either rule's recommendation. Compare release errors and unnecessary holds at the preset cutoffs and a shared acceptance rate. Proposed study in section 08; the design only.

Specify in advance how many release errors the additional evidence must prevent to justify its cost. Compare erroneous releases and unnecessary holds at the preset cutoffs and at the same acceptance rate. Report uncertainty across reporting periods. A wide uncertainty interval can leave the comparison inconclusive.

09 What a score can tell a client

A client needs to know whether the next report, deal or close will hold up. An eval score reflects the tasks, information and grading rules the system faced. Check whether the agent gathered the context it needed, carried critical steps through, used only information available at the time and performed reliably across attempts. Then test the frozen system on fresh assignments.

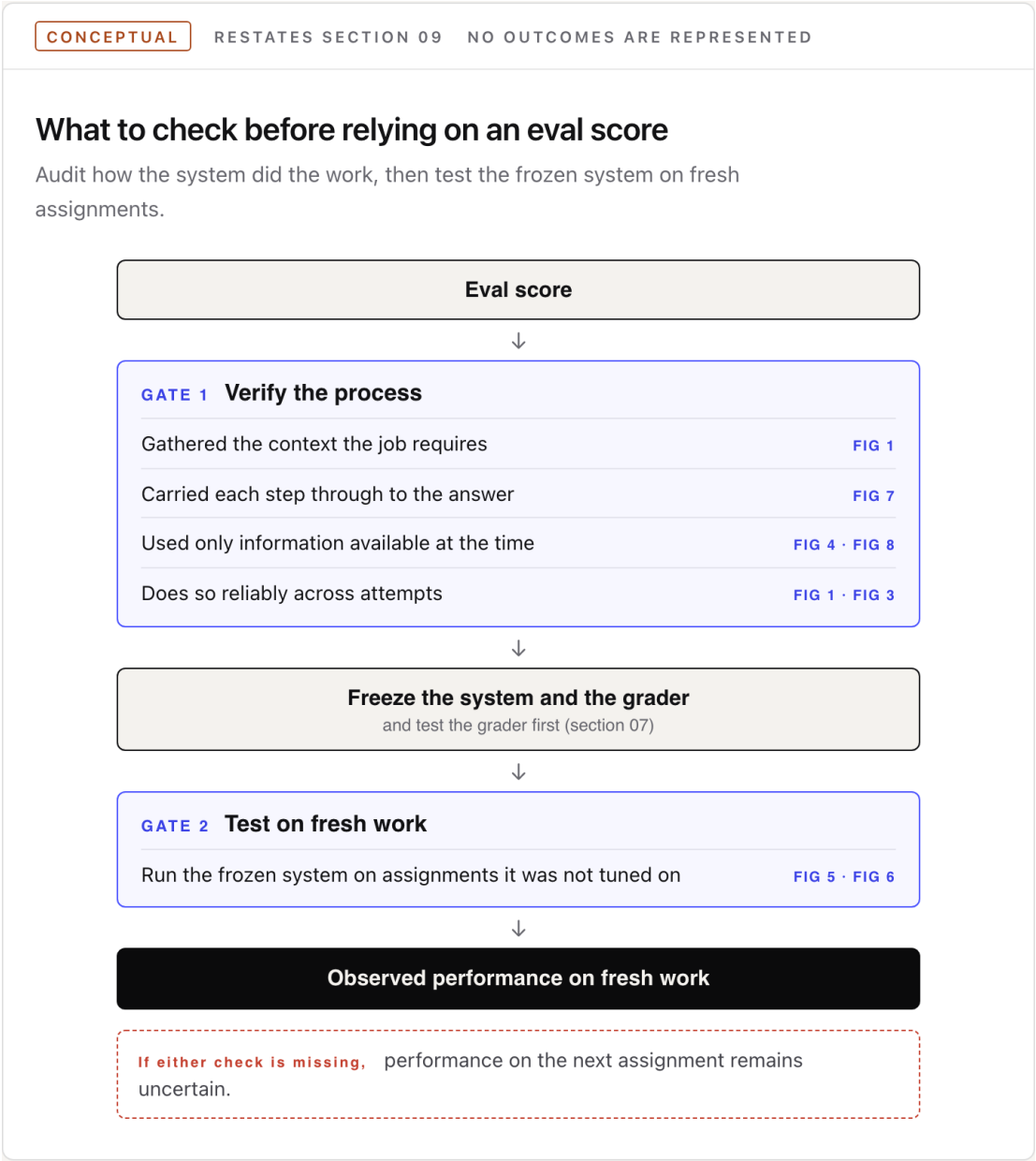


Figure 10. Conceptual. A process audit checks the conditions under which the score was earned. A fresh-work test measures the frozen system on assignments held back from development. The cited figures illustrate related risks and tests.

The proposed study tests whether process evidence improves release decisions in one workflow. How an eval score carries over to live work remains a separate question for each client.

Publication scope

This critique draws on selected public releases available by 12 November 2026. Results rely on the studies' published runs and grades. Several sources are vendor studies with small samples. Performance in client deployment remains outside the scope of the evidence reviewed here. The deadline, walk-forward and exposure examples are illustrative. The release-screening study is proposed.

Works Cited

Bailey, David H., et al. "The Probability of Backtest Overfitting." Revised working paper, 27 Feb. 2015. *eScholarship*, escholarship.org/content/qt4w1110bb/qt4w1110bb.pdf. Accessed 12 Nov. 2026. Published in *Journal of Computational Finance*, vol. 20, no. 4, 2017.

Barton, Aden. "Human Baselines for Benchmarks: AI Now Outperforms Junior Accountants." *Mercor*, 1 Oct. 2026, www.mercor.com/blog/human-baselines-for-benchmarks-ai-now-outperforms-junior-accountants/. Accessed 14 Nov. 2026.

Benchek, Julien, et al. "APEX-Accounting." *arXiv*, version 2, 30 July 2026, arxiv.org/abs/2607.27189v2. Accessed 14 Nov. 2026.

Bennett, Austin, et al. "Introducing APEX-Agents 1.1." *Mercor*, 8 Sept. 2026, www.mercor.com/blog/introducing-apex-agents-1-1/. Accessed 14 Nov. 2026.

Bertran, Martin Andres, et al. "What Fits (Into Few Tokens) Doesn't Overfit: Compression and Generalization in ML Research Agents." *arXiv*, version 1, 9 June 2026, arxiv.org/abs/2606.11045v1. Accessed 12 Nov. 2026.

Dwork, Cynthia, et al. "The Reusable Holdout: Preserving Validity in Adaptive Data Analysis." *Science*, vol. 349, no. 6248, 7 Aug. 2015, pp. 636–38, doi.org/10.1126/science.aaa9375.

Halluminate. "DealTrace: Tracing Where Frontier Models Break on a Long Private-Equity Deal Review." *Halluminate*, 17 June 2026, www.halluminate.ai/blog/dealtrace. Accessed 14 Nov. 2026.

Hyk, Alina, et al. "Westworld Finance Diligence Bench: Evaluating AI Agents on a Complete Company Acquisition Due-Diligence Process." *Halluminate*, 11 Aug. 2026, www.halluminate.ai/blog/due-diligence-bench. Accessed 14 Nov. 2026.

Kan, Max, et al. "GDPx: Measuring Frontier Model Capabilities." *SemiAnalysis*, 2 Nov. 2026, newsletter.semianalysis.com/p/gdpx-measuring-frontier-model-capabilities. Accessed 14 Nov. 2026.

Knapp Perez, Victoria. "Simulated-to-Real Transfer for Web Agents: A Small Scale Study." *Halluminate*, 27 July 2026, www.halluminate.ai/blog/sim-to-real-webagents. Accessed 14 Nov. 2026.

Li, Yiwei, et al. "Beyond Final Scores: A Systematic Evaluation of Agents for Long-Horizon AI Research and Development." *arXiv*, version 1, 13 Aug. 2026, arxiv.org/abs/2608.13417v1. Accessed 14 Nov. 2026.

Li, Zhuochun, et al. “One Success Isn’t Reliability: Thinkingbox, a Sandbox and Benchmark for Agents in Stateful Business Workflows.” *arXiv*, version 5, 31 Oct. 2026, arxiv.org/abs/2608.19741v5. Accessed 12 Nov. 2026.

OpenAI. “Why SWE-bench Verified No Longer Measures Frontier Coding Capabilities.” *OpenAI*, 23 Feb. 2026, openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/. Accessed 12 Nov. 2026.

Patwardhan, Tejal, et al. “GDPval: Evaluating AI Model Performance on Real-World Economically Valuable Tasks.” *The Fourteenth International Conference on Learning Representations (ICLR 2026)*, 2026. *arXiv*, version 1, 5 Oct. 2025, arxiv.org/abs/2510.04374v1. Accessed 14 Nov. 2026.

Tashman, Leonard J. “Out-of-Sample Tests of Forecasting Accuracy: An Analysis and Review.” *International Journal of Forecasting*, vol. 16, no. 4, 2000, pp. 437–50, doi.org/10.1016/S0169-2070(00)00065-0.

The Terminal-Bench Team. “Terminal-Bench 4.1: Updating Task Verifiers for Reward Hack Resistance and Correctness.” *Terminal-Bench*, www.tbench.ai/news/terminal-bench-4-1. Accessed 12 Nov. 2026.

Vidgen, Bertie, et al. “APEX-Agents.” *arXiv*, version 3, 23 Feb. 2026, arxiv.org/abs/2601.14242v3. Accessed 14 Nov. 2026.

Yao, Shunyu, et al. “τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains.” *arXiv*, version 1, 17 June 2024, arxiv.org/abs/2406.12045v1. Accessed 12 Nov. 2026.

Yu, Boxi, et al. “SWE-ABS: Adversarial Benchmark Strengthening Exposes Inflated Success Rates on Test-based Benchmark.” *Proceedings of the 43rd International Conference on Machine Learning*, vol. 306, 2026, pp. 149974–150011. *Proceedings of Machine Learning Research*, proceedings.mlr.press/v306/yu26b.html. Accessed 13 Nov. 2026.

Yuan, Mengqi, et al. “OSWorld 2.0: Benchmarking Computer Use Agents on Long-Horizon Real-World Tasks.” *arXiv*, version 2, 13 July 2026, arxiv.org/abs/2606.29537v2. Accessed 12 Nov. 2026.

Zheng, Pujun, et al. “SWE-Bench Pro Verified: A Reliable Benchmark for Software Engineering Agents.” *arXiv*, version 3, 9 Oct. 2026, arxiv.org/abs/2609.08149v3. Accessed 12 Nov. 2026.

Cite this essay

Aperture. “When Evals Become Overfit Backtests.” *Aperture*, 14 Nov. 2026, https://www.runaperture.com/when-evals-become-overfit-backtests.