AUDITING LLM API QUALITY BEYOND TASK SUCCESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Black-box access makes it difficult to determine whether the quality delivered by an LLM API has deteriorated. For agentic workloads, task pass rates summarize outcomes but do not reveal whether decisions were grounded in observations, instructions were followed, or errors were handled appropriately. We study whether process-quality measurements provide actionable auditing information beyond final outcomes. We propose a trajectory-based auditing framework that uses generative judges to produce dimension-specific assessments with supporting evidence and compares them with reference measurements under a controlled workload. To accommodate long trajectories, the framework combines compressed views with selective access to omitted evidence. We develop an evaluation protocol using natural trajectories, targeted quality interventions, and null edits, with independent checks of the realized effects. The protocol compares outcome-only, process-only, and combined audit signals, and evaluates detection, false alarms, evidence localization, and judging cost. Comparisons across judge models and execution harnesses assess the dependence of the measurements on the evaluator. The study examines when observable process quality complements pass rates as a practical signal for black-box API auditing.

1 INTRODUCTION

Users of a black-box language-model API see responses but have limited visibility into the model and serving configuration behind them. Users therefore need to check whether delivered quality remains comparable to a trusted reference. Existing audits examine model identity, differences from reference output distributions, and capability or safety gaps across providers (Gao et al., 2025; Zhu et al., 2025; Zhang et al., 2026b). We ask a complementary question: *does the service still perform the user's work to the expected standard?* This requires measuring task quality beyond model identity or distributional similarity.

Agentic work includes decisions and interactions with an environment. A coding assistant may pass tests despite repeated invalid tool calls, while an unsuccessful attempt may still show justified exploration and recovery. Final outcomes do not fully describe whether decisions were supported, tools were used correctly, or unnecessary work was avoided. Prior trajectory evaluations already distinguish process quality from task success (Podivilov et al., 2026; He et al., 2026).

These measurements need validation before their changes can support service auditing. A judge can repeat the same score yet miss defects or react to harmless presentation changes. The way a judge reads a record may hide a later correction or omit an observation needed to assess a decision. Scores may change with the task, execution software, or judge; reliability and judge-drift studies highlight these risks (Dev et al., 2026; Li, 2026). We therefore test repeatability and responses to independently checked edits, keeping the workload and scoring procedure fixed for service comparisons.

Our framework (Figure 1) uses a generative judge to read a task and its interaction record, then assess reasoning, tool use, efficiency, recovery from errors, and adherence to instructions, citing passages to support each score. We compare full-record prompting, compressed views with retrieval, and file-based inspection under common scoring rules. We keep the five scores separate and compare them with scores from trusted reference executions to monitor service changes. Benchmark verifier outcomes are withheld from the judge and evaluated separately to test what process scores add.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

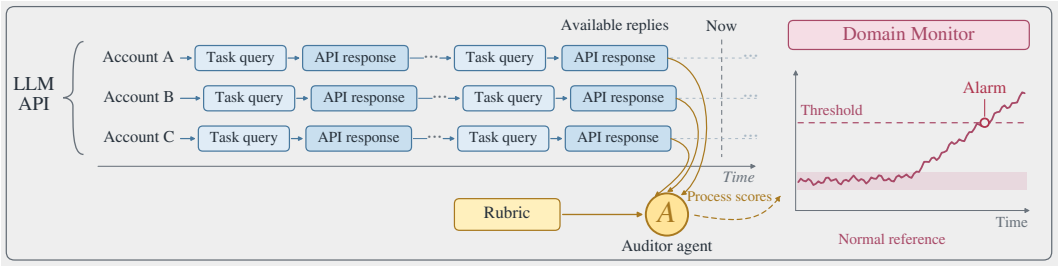


Figure 1: **Overview of online API auditing.** Across API accounts, task queries and responses accumulate over time. An auditor applies a shared rubric to available interaction records and sends process scores to an online monitor, which compares them with a normal reference. The curve illustrates an alarm when the monitoring statistic crosses a threshold; it is not an experimental result.

Across eight selected task families and 1,458 judgments, all twelve scoring configurations give lower scores to degraded records than to originals on average across their evaluated tasks, but individual edits can be missed and configurations disagree (Section 4.1). In concurrent account replays, process scores increase sensitivity to configured changes in some settings, while normal false alarms often exceed the calibration target (Section 4.2). These results do not establish an advantage at matched false-alarm rates or independently validated degradation detection.

Our contributions are:

- A process-quality measurement framework for black-box API auditing, with explicit scoring rules, applicability conditions, and supporting passages.
- A controlled validation protocol separating intended edits, independently supported quality changes, and the scoring model's response, with harmless edits as controls and checks for effects on other dimensions.
- An empirical study of repeatability and intervention sensitivity across judge models and evidence-access setups, identifying useful responses and measurement limitations.

2 Related Work

API identity and service audits. Model Equality Testing compares API and reference samples to test equality of output distributions (Gao et al., 2025), while the Rank-Based Uniformity Test uses reference-model token rankings to test behavioral equality under black-box target access (Zhu et al., 2025). AgentProv audits agentic APIs through tool-selection distributions (Wang et al., 2026); these identity statistics do not measure whether tasks were executed appropriately. Real Money, Fake Models compares third-party providers through utility, safety, and model verification (Zhang et al., 2026b). Utility mainly uses task correctness, while its generative rubric evaluator applies StrongREJECT to harmful responses. A distributional difference alone does not establish quality deterioration, and a quality alarm does not identify the backend model or its cause.

Score-based quality monitoring. Sequential risk monitoring tests whether deployed-model risk has increased, using time-uniform confidence bounds to control false alarms (Podkopaev & Ramdas, 2022); Prediction-Powered Risk Monitoring combines synthetic labels with a smaller set of true labels (Zhang et al., 2026a). BSA learns betting functions over existing behavior scores to accumulate evidence of distributional changes beyond a chosen tolerance; its main experiments use toxicity scores and BLEU (Richter et al., 2025). Who Drifted? uses rubric-based judges, sampled strong judgments, and fixed human-labeled anchors to separate system changes from judge drift in response and summary evaluation (Li, 2026). Using these detectors requires a defined auditing target and checks on reference sampling, dependence, and error control.

Trajectory evaluation and regression analysis. AgentLens reviews coding trajectories across quality dimensions and compares current runs with an anchor to diagnose regressions in complete systems, including the model, provider, agent loop, and tool interface (Podivilov et al., 2026). Its

examples include tool-parsing failures and a harness regression hidden by a normal-looking final answer. ProcCtrlBench evaluates execution processes using standardized trajectory evidence, defect categories, and control-preservation assessments (He et al., 2026). Thus, multidimensional process evaluation and trajectory-based regression detection precede our study. We focus on validating process measurements under a controlled workload and testing their value alongside final outcomes in API service comparisons. Within-trajectory monitoring predicts failure from execution prefixes, using judge-identified critical errors for supervision (Pan et al., 2026); our comparison aggregates evidence across completed executions.

Rubric-based and agentic judges. G-Eval uses criteria-guided generation and form filling to assess text quality (Liu et al., 2023); Prometheus 2 supports direct assessment and pairwise ranking under user-defined criteria (Kim et al., 2024). Agent-as-a-Judge collects evidence from artifacts and development trajectories to assess requirement satisfaction (Zhuge et al., 2025), while AgentRewardBench evaluates automatic judgments of web-agent success from trajectories (Lù et al., 2025). These works inform both what a judge assesses and how it reads evidence. We compare reading modes under common process rubrics, distinguishing process-quality scores from task-success judgments.

3 METHOD

Our framework comprises three components: evidence-based trajectory scoring, score validation, and online monitoring. A judge model scores task-execution records and cites supporting passages (Section 3.1). Score validation uses repeated judgments and controlled edits to assess repeatability and sensitivity to independently checked quality changes (Section 3.2). Online monitoring compares scores from new executions with trusted normal records to detect service changes (Section 3.3). Controlled edits are used only for offline score validation.

3.1 EVIDENCE-BASED TRAJECTORY SCORING

Task records and evidence access. To evaluate a record, the judge receives task x_t and trajectory τ_t, which contains messages, visible reasoning, and tool calls and results recorded during execution. The generating model's identity and benchmark-verifier outcomes are withheld. The judge may access the record through a full prompt, a shortened view with retrieval of omitted passages, or selective reading of a read-only file, depending on the harness the judge uses. Tool-assisted reading also appears in Agent-as-a-Judge (Zhuge et al., 2025). To support each rating, the judge cites relevant passages from the record, such as an error message together with the model's subsequent response.

Rubric-guided assessment. Following criteria-based generative evaluation (Liu et al., 2023; Kim et al., 2024), the judge scores five dimensions separately: reasoning grounding (RG), whether decisions and claims are supported by available information; tool use (TU), whether tools and arguments are used correctly; efficiency (Eff), whether unnecessary work is avoided; error recovery (ER), whether the model responds appropriately to failures; and instruction following (IF), whether it follows applicable requirements. The rubric defines when each dimension applies and what behavior warrants each integer score from 0 to 10, with higher scores indicating better quality. The judge checks instructions, consequential actions, and later corrections, then supports each score with cited passages.

Structured output. A scoring configuration specifies the model, rubric version, evidence-access mode, and execution settings. The judge returns JSON with each dimension's assessment state, score, confidence, cited passages, and explanation, plus a short overall summary. Dimensions that do not apply are marked *not applicable*; those that cannot be judged from the record are marked *insufficient evidence*. Both use `score=null`. Scores and assessment states remain separate for each dimension, without an overall numerical score.

Extraction and validation. We extract the rating JSON from the final response or the scoring software's structured result, then check required fields, whether scores agree with assessment states,

and whether cited passages exist in the record. Failed judgments and missing scores are retained, not assigned zero. These checks do not establish that the passages justify the rating. Appendix C gives the rubric boundaries, evidence-access details, output fields, and extraction rules.

3.2 Validating trajectory scores with controlled edits

In this section, we establish how we validate trajectory scores. First, for each task family, we keep the original record τ_i^O, and generate a candidate degradation τ_i^D targeting dimension k_i, and a candidate null edit τ_i^N intended to change presentation while preserving quality. The score for the targeted dimension is the *target metric*. Both edits start independently from the original. Their intended effects require validation; nulls test sensitivity to presentation alone, following principles also studied by the Judge Reliability Harness (Dev et al., 2026).

Construct and independently review edits. An editing model receives the full task and record and proposes local changes under specified constraints. We check that saved changes reconstruct each record and test local properties such as invalid tool arguments. A separate reviewer model then compares full records and changed locations in random A/B order, with model identities, original/edited labels, target dimensions, intended effects, and editor explanations hidden. It reports preference, equivalence, or uncertainty for each applicable dimension, with supporting passages. A degradation is supported when the original is preferred on its target dimension; a null is supported when all applicable dimensions are equivalent. Appendix B gives editing constraints and local checks.

Fix the panel and compare scores. We fix the selected panel before the scoring model judges each record separately, without related versions, edit locations, or review findings. Let $\bar{s}_{ik}^v$ be the mean repeated score for task family i, quality dimension k, and version $v \in \{O, D, N\}$ under one scoring configuration. We examine

$$\Delta_{ik}^D = \bar{s}_{ik}^D - \bar{s}_{ik}^O, \qquad \Delta_{ik}^N = \bar{s}_{ik}^N - \bar{s}_{ik}^O. \tag{1}$$

Supported degradations should lower the target score; nulls should produce small changes across applicable dimensions. We also examine changes in other dimensions and repeat variability: stable scores can still miss defects. Construction success (Appendix A) and sensitivity on the selected, reviewer-supported panel (Section 4.1) use different denominators.

3.3 Online monitoring

We simulate a client collecting executions through identifiable API accounts, with a hidden service change affecting a subset of accounts. Accounts maintain separate monitoring histories and alarms; tasks provide evidence within each account. We compare new executions with a *normal reference*: trusted historical records pooled across tasks (global) or from the same task (local). Inputs are verifier outcomes (V), inexpensive execution statistics such as token and tool-call counts (S), and process scores (P). Reference means and standard deviations center and scale these measurements. Gaussian maximum mean discrepancy (MMD) and a multivariate exponentially weighted moving average (MEWMA) measure departures from the reference. The judge and reference stay fixed; records enter monitoring when their scores become available.

Gaussian MMD. Within recent available records, we consider candidate times at which the service might have changed. For each candidate and task, MMD compares requests launched after that time with the normal reference (Gretton et al., 2012). The detector is applied separately to verifier, process, and execution-statistic groups. Let $R = \{r_i\}_{i=1}^n$ contain the reference vectors and W_t the w selected task vectors. Each vector contains scaled measurements and binary flags marking which measurements are available. With Gaussian kernel $k(u, v) = \exp(-\|u - v\|^2/(2\ell^2))$, we compute

$$D_t^{\mathrm{MMD}} = \frac{1}{n^2}\sum_{i,j=1}^{n} k(r_i, r_j) + \frac{1}{w^2}\sum_{u,v\in W_t} k(u, v) - \frac{2}{nw}\sum_{i=1}^{n}\sum_{u\in W_t} k(r_i, u). \tag{2}$$

This includes diagonal terms and is defined for $w = 1$; w varies with the task and candidate time. The bandwidth is fitted from the reference. The flags allow changes in which measurements are missing to contribute.

MEWMA. MEWMA accumulates departures of the mean vector from its reference value (Lowry et al., 1992). For each task and candidate, it restarts on the selected observations in score-availability order. For a fully observed standardized vector u_t,

$$z_t = (1-\lambda)z_{t-1} + \lambda u_t, \qquad D_t^{\mathrm{MEWMA}} = z_t^\top C_t^{-1} z_t, \qquad z_0 = 0. \tag{3}$$

Here λ controls how much weight each new record receives, and C_t estimates the covariance of z_t, including uncertainty in the reference mean. Missing coordinates retain their previous state, with covariance propagated according to which coordinates were observed. Appendix D specifies these updates, preprocessing, and signal combination.

Calibration and alarms. We use normal-service replays to normalize detector statistics and set account-level alarm thresholds. For each candidate change time, we aggregate evidence across tasks within the same account. Alarm rules use either a single signal type or agreement across multiple signal types. An account raises an alarm when its resulting statistic exceeds the calibrated threshold. Calibration is empirical, without a formal false-alarm guarantee. Appendix D provides the signal-combination rules and calibration procedure.

4 Experiment

Our evaluation first examines the reliability and sensitivity of trajectory judgments (Section 4.1), then tests their contribution to account-level service-change detection on a frozen trajectory collection (Section 4.2). Controlled-edit construction is evaluated separately in Appendix A.

4.1 Reliability and sensitivity of trajectory judgments

Evaluation questions and design. We test repeatability, response to supported degradations and harmless edits, agreement across scoring configurations, and agreement with human labels. Eight originals—two tasks each from AutomationBench (Shepard & Salimans, 2026), SWE-smith (Yang et al., 2025), Terminal-Bench 2.1 (Merrill et al., 2026), and ArXivMath-0626 (Dekoninck et al., 2026; MathArena, 2026)—each have one degraded and one quality-preserving null version. Grok 4.6 (SpaceXAI, 2026) supported all eight target degradations and eight nulls in a separate paired review with edit locations visible. The panel is fixed before scoring, and scoring models receive neither version labels nor review findings. Three judge models and four evidence-access setups yield 1,458 judgments, with six repetitions per scheduled record–configuration pair. Tool use is not assessed for mathematics. Appendix B details selection and six judgments recovered under a larger output budget.

A *scoring configuration* pairs a judge model with its execution software and reading procedure (*harness*). Full direct (FD) provides the full-record prompt directly with no additional harnes; compressed agentic (CA) provides a shortened view with a simple self-built retrieval harness; Codex CLI (CX) (OpenAI, 2025) and Claude Code (CC) (Anthropic, 2025) support selective file reading. Each uses the same judge models. CA covers only the two mathematics tasks and HelpScout, where compression is triggered.

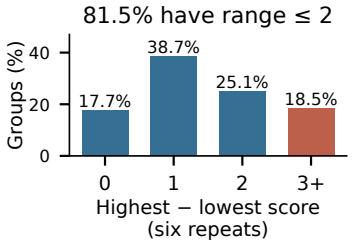


Figure 2: Highest minus lowest target score across six repeats of the same record and configuration ($n = 243$ groups).

Scores and summary measures. The target metric is the score on the degradation's intended dimension, held fixed across original (O), null (N), and degraded (D) versions. RG/TU/Eff/ER/IF identify the five dimensions defined in Section 3.1. Within each record–configuration group, we compute the six-score mean, sample standard deviation (SD), and range (highest minus lowest). SD and range measure repeat variability, whereas the mean describes assigned quality.

Compact evidence access supports more repeatable judgments on long records. On the same nine records from the three compression-eligible tasks, FD has higher mean target-score SD than

each alternative for every judge (Figure 3): 1.266 versus 0.480–0.545 for DeepSeek, 0.709 versus 0.539–0.640 for GLM, and 0.982 versus 0.519–0.862 for Qwen. This workflow comparison on three tasks shows that compression is necessary or universally sufficient in long record evaluation.

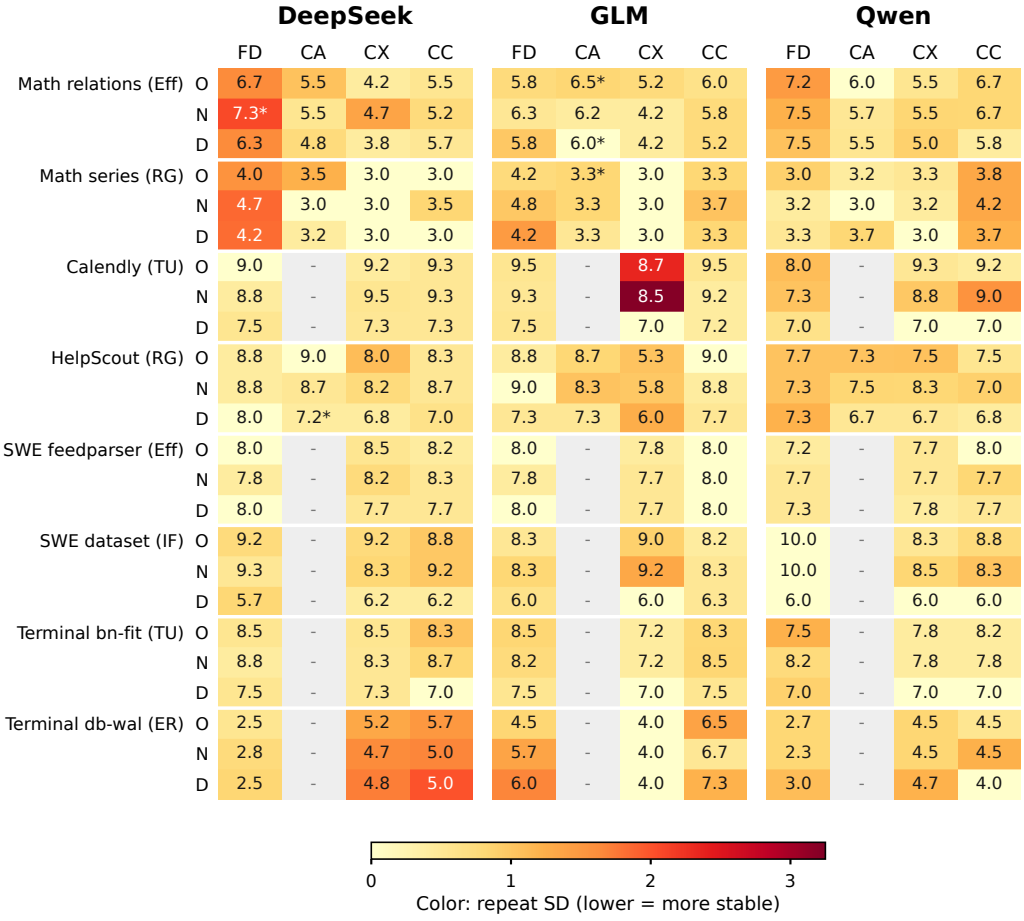


Figure 3: Scores on each task's targeted dimension, identified by RG/TU/Eff/ER/IF beside its name. O/N/D are original, null, and degraded versions. Numbers are six-repeat mean scores; color is repeat variability (SD). FD/CA/CX/CC denote the reading setups defined above. Gray – means unscheduled; * marks a larger-budget recovery. Compare scores within a task.

Repeated judgments are usually within two points. Across 243 record–configuration groups, 137 (56.4%) have a six-repeat range at most one point and 198 (81.5%) at most two; 43 are identical. Repeated judgments vary even at configured temperature 0.0.

Scores fall on average for degradations, with small changes for null edits. We use the edited-minus-original mean differences in Equation 1, holding task, target metric, and configuration fixed. Negative Δ^D indicates a response to degradation; Δ^N near zero indicates little response to the presentation edit. All twelve configurations have a negative average Δ^D on their scheduled tasks (Figure 4). Of 81 task–configuration comparisons, 57 show a decrease, 12 a tie, and 12 an increase; the overall mean is -0.821. Null changes are small: mean absolute $\Delta^N = 0.292$, with 80/81 within one point. Dataset instruction following and both tool-use interventions decrease in all nine scheduled configurations.

Some judges identify added defects without lowering the score; other cases show inconsistent attribution across dimensions (Appendix B).

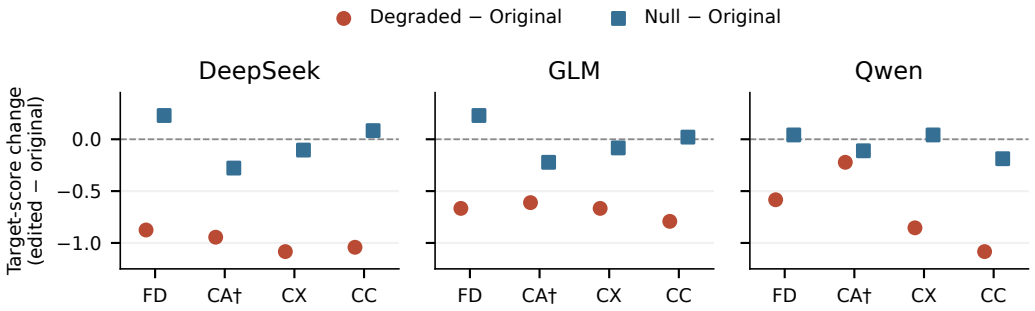


Figure 4: Edited minus original mean scores on the targeted dimension. Each point averages within-task six-repeat mean differences over eight tasks, except CA† (three tasks). Negative values mean lower ratings after editing. Unequal task sets preclude ranking harnesses.

Mean scores still differ across scoring configurations. For one record and dimension, the *mean-score gap* is the highest minus lowest six-repeat mean across configurations. Table 1 reports the largest such gap across eligible dimensions for each version; its maximizing dimension can change between O, N, and D. A dimension is eligible when every scheduled configuration has six numeric ratings. Original gaps correlate with null ($r = 0.963$) and degraded ($r = 0.668$) gaps across eight tasks (Figure 5); r denotes Pearson correlation. Comparisons holding the dimension fixed also show persistent disagreement (Appendix B).

Table 1: Largest cross-configuration mean-score gaps. Each O/N/D column selects its own maximizing dimension with complete ratings. The review dimension is the original's maximizer, not necessarily its degradation target. Assistant/human columns rate that dimension on the original. *Calendly's initial assistant score assumes the pause governs booking; its human label reflects intrinsic ambiguous timing and authority of the task.(Appendix B).

	Largest mean-score gap				Original rating	
Task	O	N	D	Review dim.	Assistant	Human
1. Calendly	6.33	6.17	4.83	IF	0*	7
2. Math series	4.33	3.50	3.83	Eff	6	6
3. Terminal db-wal	4.00	4.33	4.83	ER	4	4
4. HelpScout	3.67	3.17	2.00	RG	4	6
5. Math relations	3.00	3.33	3.67	Eff	4	6
6. Terminal bn-fit	2.83	2.83	2.33	IF	8	8
7. SWE dataset	1.83	1.67	3.17	IF	10	10
8. SWE feedparser	1.67	1.83	2.33	TU	7	7

Agreement with human labels depends on the scoring configuration. Human review assigns one dimension-specific label to each original (Table 1); Appendix B explains the three differences from initial assistant scores. Mean absolute error (MAE) averages individual ratings' absolute differences from these labels. Across nine configurations covering all eight cases, GLM-5.3/Codex has the lowest MAE (0.875 over 48 ratings; 26 exact matches), followed by GLM-5.3/full direct (1.229). Excluding db-wal, whose label uses a revised recovery rubric, leaves GLM/Codex closest. On the three reviewed cases also covered by CA, Qwen3.8-Max/CA is closest (0.500 versus 0.833 for GLM/Codex).

4.2 Online monitoring beyond task outcomes

Data and conditions. We evaluate account-level replay on sixteen tasks, four each from AutomationBench, ArXivMath-0626, Terminal-Bench 2.1, and SWE-smith, using 1,358 eligible DeepSeek trajectories scored by GLM-5.3 through Codex CLI. The judge receives neither verifier outcomes nor intervention labels. To prove the value of process quality in evaluating long records, three initial-system prompt interventions: request repeated reasoning, redundant read-only tool calls, or delayed

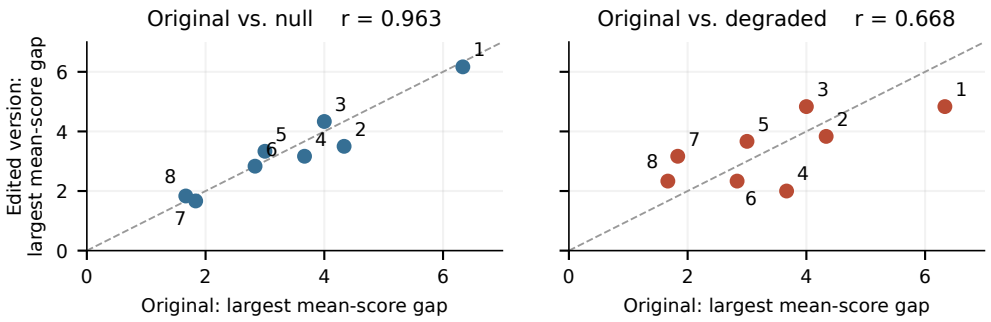


Figure 5: Largest mean-score gaps across configurations, before and after editing. Each numbered point identifies a task in Table 1. Each version selects its own maximizing dimension; points above the dashed equality line have greater disagreement after editing.

recovery are applied; the latter two apply to the twelve non-mathematics tasks.We also evaluate three deployment changes: switching the model version from V4.1 Flash to V4 Flash 0731, reducing the requested reasoning effort from high to low, and switching the serving endpoint from DeepInfra FP8 to Relace FP4. The endpoint contrast changes both the provider and the declared precision. Noticeably, these changes do not by themselves establish a decline in service quality (Appendix E).

Account replay. Each of 50 paired random splits assigns five normal trajectories per task to the reference, ten to calibration, and five to normal evaluation. We compare global and local reference schemes, with reference construction and rank calibration detailed in Appendix E. We vary account count $K \in \{1, 2, 4\}$, concurrency per account $c \in \{4, 8\}$, and affected accounts $D \in \{1, 2, 4\}$ with $D \leq K$. Five evaluation requests per task are assigned either exclusively to one account or balanced across accounts, with interleaved or later-arriving tasks. The provider selects affected accounts and a hidden switch time using normal calibration timing. Only post-switch requests on affected accounts use changed trajectories; in-flight requests retain their original condition. Each account processes its own score arrivals after recorded completion and judge latency, using the same evidence clock for all signal combinations.

Comparisons and metrics. We compare verifier-only (V), verifier and execution statistics jointly ($V \wedge S$), process-only (P), and two-of-three confirmation (Vote_2). Each component requires support from two tasks in the same account at one candidate switch time; S also requires two execution-statistic blocks. $V \wedge S$ requires both components and Vote_2 requires any two at that same candidate and update. Normal calibration targets a nominal 5% whole-account false-alarm rate. A detection requires the affected account's first alarm to follow the switch with changed evidence available; earlier alarms remain failures. All designated affected accounts remain in the denominator, including those with no changed requests. Table 2 reports macro-averaged detection and matched normal-account false alarms.

Sensitivity and false alarms trade off. For deployment changes under local MMD, Vote_2 detects 43.34% of affected accounts versus 26.28% for $V \wedge S$, while normal false alarms rise from 2.87% to 14.61%. Global P is more sensitive than V in this family, but also has higher false alarms. Prompt-intervention detection is weak or accompanied by substantial over-alarming; combined global detection is near zero. These results show sensitivity under different decision rules, not an advantage at matched realized false-alarm rates. Condition and account-matrix results appear in Appendix E.

Calibration and interpretation remain limited but also very promising. Few independent normal trajectories constrain tail calibration; repeated splits do not add independent data. Rank and threshold calibration use separate replays of the same small calibration pool. A disjoint-source diagnostic largely eliminates local alarms by saturating thresholds, rather than preserving detection (Appendix E.5). Interventions may not cause observable defects, which can limit detection against

Table 2: Account-level detection / normal false-alarm rates (%). Bold marks the highest detection rate in each row. Each family averages equally over twelve valid (K, c, D) settings, three conditions, both assignment strategies, and both arrival layouts, with fifty splits per setting. Normal streams are reused for matching, not counted as independent evidence. V, S, and P denote verifier, execution statistics, and process scores; Vote_2 requires at least two components. Nominal $\alpha = 0.05$ is not an achieved error guarantee.

Detector	Reference	V	$V \wedge S$	P	Vote_2
Prompt interventions					
MEWMA	Global	**2.44** / 3.11	0.01 / 0.07	2.25 / 3.90	0.01 / 0.18
MEWMA	Local	10.29 / 11.98	1.75 / 2.23	**28.23** / 33.06	9.09 / 9.13
MMD	Global	0.31 / 0.60	0.00 / 0.00	**2.70** / 5.18	0.11 / 0.29
MMD	Local	11.07 / 12.05	1.59 / 2.17	**28.64** / 34.64	10.16 / 9.24
Deployment changes					
MEWMA	Global	2.43 / 2.97	0.29 / 0.09	**9.00** / 6.82	2.03 / 0.47
MEWMA	Local	33.82 / 17.00	27.16 / 2.90	43.12 / 45.50	**43.36** / 14.13
MMD	Global	1.33 / 1.29	0.02 / 0.02	**12.87** / 7.43	1.31 / 0.40
MMD	Local	33.65 / 17.14	26.28 / 2.87	41.47 / 47.14	**43.34** / 14.61

condition labels; this cannot explain false alarms on wholly normal streams. However, current experiment results do reveal the potential of process quality in evaluating long records, especially in terms of consistent online auditing.

5 Limitations and Future Work

The five dimensions cover selected aspects of observable execution, not a comprehensive assessment of process quality or model capability. Our small controlled-edit panel does not establish transfer to broader workloads or natural failures. Benchmark verifiers cover only checked outcomes, leaving tasks without reliable verifiers less supported. Scores depend on the judge, rubric, and evidence access (Section 4.1). Repeatability does not establish correct ratings or explanations, and our small, non-blind human review offers limited calibration evidence. Evaluator changes require comparability checks before score shifts are interpreted as service deterioration (Li, 2026). Future work should broaden task and dimension coverage with independent, blinded review of ratings and supporting evidence.

Online monitoring remains exploratory: few independent normal trajectories limit calibration, and reliable false-alarm control remains unestablished (Section 4.2; Appendix E.5). Prompt and deployment changes need not cause quality loss, so condition detection alone does not establish degradation detection. This uncertainty may explain false alarms on normal streams to some extent. Simulated accounts and recorded durations do not establish actual provider targeting or real deployment latency. Judging long records adds token and latency costs that depend on workload and scoring policy; no general cost advantage is established. Future work should validate realized defects, calibrate on larger independent normal collections, and test more efficient evidence selection and selective scoring while checking detection and false alarms.

AI use statement

We used generative AI tools to assist with implementing experimental methods and analysis code, interpreting experimental results, checking cited literature, preparing figures, and drafting and revising the manuscript. Language models also served as experimental components: they generated task trajectories and controlled edits, reviewed edits, and produced trajectory-quality assessments. Their roles and the limits of model-based validation are described in Sections 3 and 4. Reported numerical summaries were checked against saved experiment artifacts; selected trajectory judgments were examined in an assistant-supported, non-blind human review (Appendix B). The authors have reviewed all AI-assisted work and take responsibility for the final text, claims, code, and other artifacts, including those developed with AI assistance.

Ethics Statement

This work studies the auditing of task-execution quality in black-box LLM APIs. Our experiments use existing research benchmarks, controlled trajectory edits, and replayed service streams. Behavioral interventions are applied within benchmark experiments; prompts involving repeated tool calls restrict repetition to safe operations. The limited human review is assistant-supported and non-blind, and is reported as development evidence rather than independent ground truth.

Automated judgments may reflect evaluator biases, incomplete evidence, or rubric ambiguity. An audit alarm therefore does not by itself establish quality degradation, model substitution, or provider misconduct. Our framework is intended to support further investigation, and consequential decisions should require corroborating evidence. We report evaluator disagreement, unsuccessful interventions, and false alarms to make these limitations explicit.

Reproducibility Statement

The main paper describes the scoring framework, controlled-edit protocol, and monitoring methods. Appendices A and B document edit construction, panel selection, repeated judgments, and recovery policies. Appendix C specifies the assessment dimensions, evidence-access modes, and output-validation rules. Appendices D and E describe feature processing, detector settings, data partitions, replay scheduling, threshold calibration, and condition-level results.

We present configuration and prompt, edit provenance, raw judging attempts, and evidence references. The experiments distinguish source trajectories from repeated judgments and replay streams, and document exclusions and changes to scoring policies. Because API serving conditions and agentic execution can vary, identical settings may not produce identical trajectories or ratings. Reproducing the reported analyses therefore requires distinguishing analysis of the frozen records from new API executions.

References

Alibaba Cloud. Qwen3.8-Max. Model documentation, 2026. URL https://github.com/AlibabaCloud-Official/Qwen3.8-max. Accessed September 23, 2026.

Anthropic. Claude Code. Software repository, 2025. URL https://github.com/anthropics/claude-code. Accessed September 23, 2026.

DeepSeek-AI. Introducing DeepSeek-V4.1-Flash: Smarter, faster, more efficient. Model release announcement, September 2026a. URL https://deepseek.com/en/news/deepseek-v4-1-flash/. Accessed September 23, 2026.

DeepSeek-AI. DeepSeek-V4-Flash. Model card, 2026b. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash. Accessed September 23, 2026.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: MathArena as an evaluation platform for mathematics with LLMs. *arXiv preprint arXiv:2605.00674*, 2026. URL https://arxiv.org/abs/2605.00674v2. Version 2.

Sunishchal Dev, Andrew Sloan, Joshua Kavner, Nicholas Kong, and Morgan Sandler. Judge reliability harness: Stress testing the reliability of LLM judges. *arXiv preprint arXiv:2603.05399*, 2026. URL https://arxiv.org/abs/2603.05399v1. Accepted to the ICLR 2026 Agents in the Wild workshop.

Irena Gao, Percy Liang, and Carlos Guestrin. Model equality testing: Which model is this API serving? In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.20247v2.

Google. Gemma 4 31B IT. Model card, 2026. URL https://huggingface.co/google/gemma-4-31B-it. Accessed September 23, 2026.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL https://jmlr.org/papers/v13/gretton12a.html.

Jiawei He, Jie Jia, Chenbo Liu, Chaoyi Xue, Yapeng Song, Xikai Yang, and Dong Sun. ProcCtrl-Bench: Evaluating process-level defects and control preservation in LLM coding agents. *arXiv preprint arXiv:2605.20251*, 2026. URL https://arxiv.org/abs/2605.20251v4. Version 4.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.248. URL https://aclanthology.org/2024.emnlp-main.248/.

Yitao Li. Who drifted: the system or the judge? anytime-valid attribution in LLM evaluation pipelines. *arXiv preprint arXiv:2606.15474*, 2026. URL https://arxiv.org/abs/2606.15474v1.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Cynthia A. Lowry, William H. Woodall, Charles W. Champ, and Steven E. Rigdon. A multivariate exponentially weighted moving average control chart. *Technometrics*, 34(1):46–53, 1992. doi: 10.1080/00401706.1992.10485232. URL https://doi.org/10.1080/00401706.1992.10485232.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. AgentRewardBench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*, 2025. URL https://arxiv.org/abs/2504.08942v2. Version 2.

MathArena. ArXivMath-0626. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/MathArena/arxivmath-0626. June 2026 release. Accessed September 23, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026. URL https://arxiv.org/abs/2601.11868v1. Dataset release 2.1: https://github.com/harbor-framework/terminal-bench-2-1.

Meta. Introducing Muse Spark 1.3. Model release announcement, September 2026. URL https://research.meta.ai/blog/introducing-muse-spark-1-3. Accessed September 23, 2026.

OpenAI. Codex CLI. Software repository, 2025. URL https://github.com/openai/codex. Accessed September 23, 2026.

Sitong Pan, Yipeng Shen, Yilin Lu, Caiwen Ding, Lu Cheng, and Qianwen Wang. Monitoring web agents without internal signals: Observable trajectories and key-step supervision. *arXiv preprint arXiv:2609.02057*, 2026. URL https://arxiv.org/abs/2609.02057v1.

Andrey Podivilov, Vadim Lomshakov, Sergey Savin, Matvei Startsev, Roman Pozharskiy, Maksim Parshin, and Sergey Nikolenko. AgentLens: Production-assessed trajectory reviews for coding agent evaluation. *arXiv preprint arXiv:2607.06624*, 2026. URL https://arxiv.org/abs/2607.06624v2. Version 2.

Aleksandr Podkopaev and Aaditya Ramdas. Tracking the risk of a deployed model and detecting harmful distribution shifts. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.06177v4.

Leo Richter, Xuanli He, Pasquale Minervini, and Matt J. Kusner. An auditing test to detect behavioral shift in language models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7af140f516c9f57050f7359bf53bc8f0-Abstract-Conference.html.

Daniel Shepard and Robin Salimans. AutomationBench. *arXiv preprint arXiv:2604.18934*, 2026. URL https://arxiv.org/abs/2604.18934v1.

SpaceXAI. Introducing Grok 4.6. Model release announcement, August 2026. URL https://x.ai/news/grok-4-6. Accessed September 23, 2026.

Tencent Hy Team. Hy3. Model card, 2026. URL https://huggingface.co/tencent/Hy3. Accessed September 23, 2026.

Xun Wang, Bihe Zhao, Michael Backes, Franziska Boenisch, and Adam Dziedzic. AgentProv: Auditing agentic LLM API providers via tool-use policy probes. *arXiv preprint arXiv:2609.00052*, 2026. URL https://arxiv.org/abs/2609.00052v1. Accepted to EMNLP 2026.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025. URL https://arxiv.org/abs/2504.21798v2. Version 2.

Z.ai. GLM-5.3. Model card, 2026. URL https://huggingface.co/zai-org/GLM-5.3. Accessed September 23, 2026.

Guangyi Zhang, Yunlong Cai, Guanding Yu, and Osvaldo Simeone. Prediction-powered risk monitoring of deployed models for detecting harmful distribution shifts. *arXiv preprint arXiv:2602.02229*, 2026a. URL https://arxiv.org/abs/2602.02229v2. Version 2; accepted to ICML 2026.

Yage Zhang, Yukun Jiang, Zeyuan Chen, Michael Backes, Xinyue Shen, and Yang Zhang. Real money, fake models: Deceptive model claims in shadow APIs. *arXiv preprint arXiv:2603.01919*, 2026b. URL https://arxiv.org/abs/2603.01919v3. Version 3; accepted to ACM CCS 2026.

Xiaoyuan Zhu, Yaowen Ye, Tianyi Qiu, Hanlin Zhu, Sijun Tan, Ajraf Mannan, Jonathan Michala, Raluca Ada Popa, and Willie Neiswanger. Auditing black-box LLM APIs with a rank-based uniformity test. *arXiv preprint arXiv:2506.06975*, 2025. URL https://arxiv.org/abs/2506.06975v5. Version 5, revised April 2026.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80569–80611. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhuge25a.html.

A VALIDATION OF CONTROLLED EDITS

This preliminary study evaluates edit construction on 40 originals. The separately selected panel used for the main sensitivity experiment and its review limitations are described in Appendix B.

We test whether controlled edits realize their prescribed effects: a degradation should reduce a designated quality metric, while a null edit should preserve all applicable metrics. We selected 40 source trajectories from the frozen extra120 panel, spanning eight tasks, two each from AutomationBench, SWE-smith, Terminal-Bench 2.1, and ArXivMath-0626 (benchmark references in Section 4.1). The panel contains eight trajectories from each of five sources: DeepSeek V4 Flash (DeepSeek-AI, 2026b), Gemma 4 31B (Google, 2026), HY3 (Tencent Hy Team, 2026), Muse Spark 1.3 (Meta, 2026), and GLM-5.3 (Z.ai, 2026). Each trajectory defines a family with one planned degradation and one planned null, yielding 80 candidate comparisons. Eight families target each of reasoning grounding, tool use, efficiency, error recovery, and instruction following; tool use is inapplicable to mathematics. The DeepSeek Flash API (`deepseek-flash`) generated edits at at most one location per candidate, allowing an inserted tool-call cycle at that location. Nulls comprise paraphrases, layout changes, and formula-formatting changes. Final answers are protected for all nulls and for degradations except those targeting instruction following.

Grok 4.6 (SpaceXAI, 2026) independently reviewed the complete A/B versions with edit locations provided. A/B order was determined by a fixed seed, and source identities, original/candidate identities, edit strategies, and intended directions were hidden. The editor used high reasoning effort with temperature omitted; the reviewer used medium effort and temperature zero. For each applicable metric, the reviewer returned a preference, equivalence, or uncertainty, with quotations and contextual justification. We count a degradation as supported when the original is preferred on the target metric, and a null as supported when all applicable metrics are equivalent. We retain the first valid review for each candidate, including unsupported judgments; recovery fills only missing valid results. Of 80 planned candidates, 74 were generated and reviewed: 38 degradations and 36 nulls. Four candidates failed generation, and two were inapplicable because the only remaining edit location was the protected final answer.

Table 3: Validation support for controlled edits. N is the number of generated candidates, L the number supported by the reviewer, D the number with a decidable programmatic local effect, P the number programmatically supported, and U the number supported by either method, counted once per candidate. Programmatic support concerns the prescribed local effect.

Edit	Generated / planned	LLM L/N	Programmatic checks Support P/D	Programmatic checks Coverage D/N	Either method U/N
Degradation	38/40	26/38 (68.4%)	8/8	8/38	26/38 (68.4%)
Null	36/40	36/36 (100%)	16/16	16/36	36/36 (100%)
All	74/80	62/74 (83.8%)	24/24	24/74	62/74 (83.8%)

Complementary offline checks examined the same frozen candidates without additional API calls. They supported prescribed local effects in 24/74 cases (32.4% coverage): eight tool-argument contract violations, thirteen text-layout nulls, and three formula-formatting nulls. All 24 decidable cases were supported; the other 50 were deferred. These checks establish local properties within their parsing and semantic scope. Tool feedback was not replayed in an environment, and mathematical checks cover only supported local expressions. Reconstruction and protected-content checks alone do not establish degradation of a quality metric.

Table 3 shows that the reviewer supported 26/38 degradations (68.4%) and all 36 available nulls. All programmatically supported cases also received reviewer support, so support by either method remains 62/74 (83.8%), or 62/80 (77.5%) over all planned candidates. The corresponding planned-candidate rates are 26/40 (65.0%) for degradations and 36/40 (90.0%) for nulls. Degradation support varies across target metrics: tool use achieves 8/8, reasoning grounding 6/7, instruction following 5/7, efficiency 4/8, and error recovery 3/8. All twelve unsupported degradations were judged equivalent on the target metric. Programmatic evidence corroborates the tool-contract violations and sixteen formatting nulls; the twenty paraphrase nulls rely on model-based judgments of semantic preservation.

Targeted edits also affect other dimensions: 14 of the 26 supported degradations changed another metric according to the reviewer. Of 35 families with both candidates available, 23 supported both target degradation and null equivalence (65.7%; 23/40, or 57.5%, of planned families). In ten of these families, the reviewer also judged every other applicable metric equivalent under degradation; this subset was selected after validation. Local defects likewise need not imply the intended metric change: in one error-recovery edit, a program detected a new false arithmetic equality, but the reviewer judged the target metric equivalent. The local finding was not counted as support for recovery degradation. We therefore retain local evidence, target-metric judgments, and other metric changes separately. The rates are descriptive: families share eight underlying tasks, quality judgments come from one independent reviewer without human calibration, and the null results concern candidates accepted under generation constraints. These findings characterize the controlled-edit set; judge sensitivity and API-auditing gains require separate evaluation.

B CONTROLLED-PANEL CONSTRUCTION AND ANALYSIS DETAILS

Edit construction and local checks. For each original, the edit request specifies the target dimension and strategy, allowed locations, maximum number of edited locations, and content that must remain unchanged. The editing model receives the full task and record and proposes replacements or, where permitted, an additional tool call and result. Degradations can introduce unsupported reasoning, invalid tool arguments, unnecessary work, poor recovery, or instruction violations. Separate null requests use paraphrasing, layout changes, or formula formatting; the two edits need not share locations or size. We check edit locations and count, protected content, tool-call/result pairing, and exact reconstruction of the edited record from the saved changes. Additional checks establish decidable local properties, such as invalid tool arguments, preserved formatting, or arithmetic errors. A confirmed local defect need not impair the targeted quality dimension: an arithmetic error, for example, need not impair recovery. Unresolved checks, failed generation, unavailable reviews, and unsupported edits remain recorded in construction statistics.

Selection and validation scope. The main study uses one original per task, selected separately from the 40-original construction study in Appendix A. The eight degradations target relation-type mathematics (efficiency), mathematics series (reasoning grounding), Calendly (tool use), HelpScout (reasoning grounding), feedparser (efficiency), dataset (instruction following), bn-fit-modify (tool use), and db-wal-recovery (error recovery). Their realized edit counts are 1, 1, 2, 4, 3, 2, 1, and 2. Nulls preserve final answers; degradations preserve final answers except when targeting instruction following. Preserving a recorded answer does not establish an unchanged verifier outcome after intervention.

Relation-type mathematics and db-wal use earlier reviewed degradations; the other six use the fifth generation version. Grok saw complete A/B records, changed locations, and tool definitions, with source identities, original/candidate identities, target metric, intended direction, and editor explanations hidden. Its review was used to select versions before pointwise scoring. The eight supported degradations and eight equivalent nulls characterize this selected panel, not generation success rates or scalar gold scores. Scoring models were not told which passages had been edited.

All 16 variants pass reconstruction checks. Programmatic checks support the local effects of two degradations and five nulls; six degradations and three nulls remain deferred. Five degradations have reviewer-reported cross-metric changes and three have reviewer-reported new contextual inconsistencies; db-wal also has a manually noted plan/action inconsistency. The relation-type efficiency insertion explicitly describes its calculation as redundant. Synthetic tool feedback was not replayed in a live environment. These limitations preclude interpreting the panel as perfectly isolated or uniformly subtle interventions.

Completed snapshot and output-policy recovery. The frozen result index contains 1,458 successful logical judgments: 432 each for full direct, Codex, and Claude Code, and 162 for compressed agentic. The models are DeepSeek V4.1 Flash (DeepSeek-AI, 2026a), GLM-5.3 (Z.ai, 2026), and Qwen3.8-Max (Alibaba Cloud, 2026); CLI names designate harnesses using these same models. One logical judgment is one sample, model, harness, and repeat index. Retry results replace failed positions rather than creating new repetitions. No score-dependent replacement is performed.

Six builtin positions required a later recovery policy: four GLM compressed-agentic mathematics judgments, one DeepSeek full-direct mathematics judgment, and one DeepSeek compressed-agentic HelpScout judgment. This policy retains the initial request, adds a fixed continuation reminder after output-length termination, and permits 262,144 cumulative output tokens with at most 65,536 per request. Unlike the earlier incomplete snapshot, the current main analysis includes these six successful replacements and marks their five affected sample–condition groups. Four target-metric O/N/D triplets contain a replacement. Excluding those entire triplets leaves 54/77 decreases (70.1%) and mean $\Delta^D = -0.829$, close to 57/81 (70.4%) and -0.821 in the completed snapshot. It does not establish that output policies are interchangeable.

CLI retries also involved session recovery, infrastructure fixes, and scheduling changes. Temperature is configured as zero, but routing, reasoning controls, and tool interactions differ across harnesses. The records measure realized judging workflows and do not isolate decoding, server-side, or harness contributions to variability.

Numeric completeness and paired contrasts. There are 1,143 scheduled sample–condition–metric groups, excluding mathematics tool use. Of these, 1,126 have all six numeric scores; 34 individual error-recovery ratings are not applicable, distributed across 17 groups with changing applicability. These groups are not filled with zeros or used as complete six-score estimates. All 243 target-metric groups and all 81 target-metric triplets are complete. The eight underlying tasks, not the 24 variants or 1,458 judgments, determine the independent case count.

Table 4 gives target-metric means and contrasts on each harness's scheduled families. Negative contrast counts are descriptive rather than significance tests. Null MAD (mean absolute difference from the original) is $\mathrm{mean}\,|\bar{s}^N - \bar{s}^O|$ on the same target metric; it is not an API-level false-alarm rate. The aggregate means are condition-weighted, so compression-eligible tasks occur under more configurations. For strict four-harness comparisons we restrict all harnesses to the same nine objects from the three compression-eligible tasks.

Table 4: Target-metric sensitivity on scheduled tasks. $\downarrow /n$ counts tasks with negative degraded–original mean differences. CA has three tasks and the other harnesses eight. Null MAD uses target metrics, not all metrics pooled.

Judge	Harness	$\downarrow /n$	O	N	D	Δ^D	Null MAD
DeepSeek	FD	5/8	7.08	7.31	6.21	-0.88	0.31
DeepSeek	CA	3/3	6.00	5.72	5.06	-0.94	0.28
DeepSeek	CX	7/8	6.96	6.85	5.88	-1.08	0.35
DeepSeek	CC	6/8	7.15	7.23	6.10	-1.04	0.33
GLM	FD	4/8	7.21	7.44	6.54	-0.67	0.40
GLM	CA	2/3	6.17	5.94	5.56	-0.61	0.22
GLM	CX	5/8	6.27	6.19	5.60	-0.67	0.25
GLM	CC	5/8	7.35	7.38	6.56	-0.79	0.19
Qwen	FD	4/8	6.65	6.69	6.06	-0.58	0.37
Qwen	CA	2/3	5.50	5.39	5.28	-0.22	0.22
Qwen	CX	6/8	6.75	6.79	5.90	-0.85	0.21
Qwen	CC	8/8	7.08	6.90	6.00	-1.08	0.27

height

Table 5: Mean target-metric sample SD over the same nine compression-eligible objects for each judge. Every cell averages nine six-repeat groups, including marked output-policy recoveries. Lower variability does not imply better grounding or sensitivity.

Judge	Full direct	Compressed	Codex	Claude Code
DeepSeek	1.266	0.545	0.493	0.480
GLM	0.709	0.640	0.539	0.636
Qwen	0.982	0.519	0.621	0.862

height

Direction, severity, and score anchoring. Across the 81 target-metric triplets, 30 decrease by at least one point, 50 place degraded below both original and null, and only 19 completely separate

all six degraded scores below all six original scores. Thus an aggregate decrease should not be presented as reliable single-call classification. The mathematics-series case remains approximately unchanged on grounding (mean $\Delta^D = +0.014$), and db-wal on recovery ($+0.148$). For mathematics series, DeepSeek/Codex assigns grounding $3 \to 3$: it identifies additional local errors but retains the original's low score for an unsupported global conclusion. In db-wal, selected rationales inconsistently attribute faults to efficiency and recovery. These cases do not justify attributing every non-decrease to cautious scoring.

Cross-configuration score gaps and completeness. For family i, version v, dimension k, and configuration c, let $\bar{s}^v_{ikc}$ be the six-repeat mean. The mean-score gap is $\max_c \bar{s}^v_{ikc} - \min_c \bar{s}^v_{ikc}$. The main table takes the largest gap across dimensions having six numeric ratings under every scheduled configuration. This is what completeness means here; it does not mean that all five dimensions must apply. Missing or not-applicable ratings are not replaced by zeros. The maximizing dimension can change: mathematics-series original selects efficiency, whereas null and degraded select recovery. Fixing each task's metric to the original maximizer instead gives original–null/degraded Pearson correlations of 0.941/0.810. Across all 35 strictly matched task–metric groups, the corresponding correlations are 0.933/0.792; using only the nine configurations common to every task gives 0.928/0.793. Excluding any condition containing a new-budget result from all three variants gives 0.933/0.792. Removing Calendly from the eight task-maximum comparisons reduces the correlations to 0.905/0.481, showing that the degraded association partly depends on this case. No significance or universal calibration claim follows from these dependent, selected observations.

Review scope and evidence access. Initial assistant reviews inspect the frozen original records and metric rules with existing disagreements known: Astra assesses the first four cases, and a later assistant assesses the remaining four. Human review uses these discussions and is neither blinded nor independent of them. Each label concerns the original's dimension with the largest mean-score gap, rather than necessarily the edit's target dimension. For example, Calendly's edit targets tool use, but its largest original gap is in instruction following. All eight human labels are now confirmed; initial assistant scores remain separate. Three cases have different assistant and human scores.

For Calendly instruction following, the initial assistant score is 0 and the human label is 7. The initial assessment treats the booking pause as still governing the action. The human review finds the timing and booking-policy authority too ambiguous to establish a governing authorization violation, but retains a deduction for the final summary's explanation of discarded guidance, which violates a local reporting constraint.

For HelpScout grounding, the initial assistant score is 4 and the human label is 6. The initial assessment treats the report's counting error and unsupported outreach claims as defects in separate important branches. The human review finds that reply-count and total-message terminology affect the report, but their effect on tagging or outreach recipients is not established. Unsupported claims of completed engineering arrangements still affect the outreach stage, supporting the narrower scope of the 6-point anchor.

For mathematics-relations efficiency, the initial assistant score is 4 and the human label is 6. The initial assessment gives greater weight to speculative branches and answer redrafting. The human review counts reasonable attempts to prove the necessary upper bound as useful work, even when unsuccessful. The deduction concerns repeated side investigations without new evidence; the record supports an extended avoidable branch, but does not establish that it outweighs the useful main approach.

The remaining five human labels match the initial assistant scores. Mathematics-series efficiency is 6for avoidable repeated derivation branches; this does not certify a global proof. Db-wal recovery is 4for incomplete handling of the newly missing WAL and repeated search failures, under the revised scope described below. Bn-fit instruction following is 8for missing step explanations despite compliant main steps and outputs; the task uses SHOULD for explanations and MUST for bash calls. Dataset instruction following is 10for the required reproduction, one-line non-test fix, checks, cleanup, and submission; the one-call requirement applies only after it appears. Feedparser tool use is 7for locally retried calls and a corrected Atom check, without evidence that these defects disrupt dependent work or shared state. Only the confirmed human labels enter the alignment comparisons.

Recovery scope revised after the pilot. Manual inspection prompted a change to the agent error-recovery rubric within the existing template version. A fault that the task explicitly asks the agent to repair is no longer, by itself, a recovery opportunity; unexpected execution failures or evidence invalidating an adopted approach are. The revision also makes supported stopping explicit in intermediate anchors. The db-wal human label uses this revised scope, whereas all reported judge ratings retain their earlier frozen prompts. No stability experiment was rerun. Alignment differences involving this label may therefore reflect both rubric scope and judgment; the old stability results do not validate the revised prompt. We retain the original and revised recovery text and their hashes with the paper assets and report alignment excluding this case.

Alignment with the final human labels. For a matched set of K reviewed cases, we use $\mathrm{MAE} = (6K)^{-1} \sum_{i=1}^{K} \sum_{r=1}^{6} |s_{ir} - y_i|$, where y_i is the human label for the case's selected metric. This averages individual-rating errors with equal case weights; it does not allow errors above and below a label to cancel through a six-repeat mean. Table 6 compares the nine configurations covering all eight cases. Computing absolute error after averaging each case's ratings also places GLM-5.3/Codex first (0.875). Excluding db-wal leaves seven cases and 42 ratings per configuration; GLM-5.3/Codex still has the lowest MAE (1.000), followed by Qwen3.8-Max/full direct (1.286). These are descriptive comparisons on selected cases, not population accuracy estimates or independent samples of size 48.

Table 6: Human alignment on the same eight originals, sorted by individual-rating MAE. Case columns show six-repeat means; human labels are shown in the header. Exact counts ratings equal to the human label. Case columns are Calendly, math series, db-wal, HelpScout, math relations, bn-fit, dataset, and feedparser, respectively. Harness abbreviations follow Section 4.1.

Judge	Harness	Cal. (7)	Series (6)	WAL (4)	Help (6)	Rel. (6)	BN (8)	Data (10)	Feed (7)	MAE	Exact
GLM	CX	5.00	5.00	4.00	5.33	5.17	9.50	9.00	7.00	**0.875**	26/48
GLM	FD	5.17	5.67	4.50	8.83	5.83	9.67	8.33	7.17	1.229	18/48
Qwen	FD	7.67	4.00	2.67	7.67	7.17	7.17	10.00	8.33	1.292	13/48
DeepSeek	CC	5.50	5.67	5.67	8.33	5.50	9.67	8.83	8.67	1.396	11/48
DeepSeek	CX	4.17	5.67	5.17	8.00	4.17	9.17	9.17	8.00	1.438	11/48
Qwen	CC	3.33	6.83	4.50	7.50	6.67	9.83	8.83	7.67	1.438	11/48
GLM	CC	5.50	6.17	6.50	9.00	6.00	10.00	8.17	7.17	1.521	18/48
Qwen	CX	1.33	6.67	4.50	7.50	5.50	9.33	8.33	7.33	1.521	15/48
DeepSeek	FD	5.17	8.33	2.50	8.83	6.67	8.00	9.17	8.00	1.625	5/48
height											

Table 7: All twelve configurations on the same three compression-eligible reviewed cases: Series (math series), Help (HelpScout), and Rel. (math relations). Qwen/compressed has the lowest MAE here; this three-case result is separate from the eight-case ranking. GLM/compressed mathematics includes the frozen new-budget replacement described above.

Judge	Harness	Series (6)	Help (6)	Rel. (6)	MAE	Exact
Qwen	CA	5.83	7.33	6.00	**0.500**	13/18
GLM	CX	5.00	5.33	5.17	0.833	9/18
Qwen	CX	6.67	7.50	5.50	0.889	7/18
Qwen	CC	6.83	7.50	6.67	1.111	4/18
DeepSeek	CC	5.67	8.33	5.50	1.167	5/18
GLM	CC	6.17	9.00	6.00	1.167	9/18
GLM	FD	5.67	8.83	5.83	1.222	7/18
GLM	CA	6.33	8.67	6.50	1.278	5/18
DeepSeek	CX	5.67	8.00	4.17	1.500	4/18
DeepSeek	CA	5.00	9.00	5.50	1.611	3/18
Qwen	FD	4.00	7.67	7.17	1.611	4/18
DeepSeek	FD	8.33	8.83	6.67	2.167	2/18
height						

Earlier execution diagnostics found tool-output truncation in some Codex readings, incomplete fragment coverage in some compressed-agentic judgments, and full-direct rationales contradicting supplied reasoning. A complete evidence file and a valid score JSON do not guarantee faithful evidence

Table 8: Process-quality dimensions and key rubric boundaries. Higher scores indicate better assessed quality within each dimension.

Dimension	Assessment target	Key boundary
Reasoning grounding	Support for plans, decisions, diagnoses, and claims in task facts, observations, and valid inference.	A tested hypothesis differs from an unsupported premise adopted to guide later work.
Tool use	Tool selection, arguments, preconditions, ordering, and handling of execution state.	A valid call can still violate instructions; a repaired execution error still affects this dimension.
Efficiency	Whether work addresses a task need or an unresolved question given the information then available.	Justified exploration and required checks are useful; length or call count alone does not establish waste.
Error recovery	Recognition, diagnosis, adaptation, and verification after a failure or contradiction.	A supported stop or escalation can be appropriate; no recovery opportunity is not a perfect recovery score.
Instruction following	Compliance with explicit requirements on authorization, scope, methods, required steps, and response form.	Requirements apply to the actions or replies they govern; optional reasonable choices are allowed.

use. These are case diagnostics, not a measured semantic-error rate across the completed panel. We retain the recorded scores and provenance rather than removing disagreements with intended edits.

C Trajectory scoring details

C.1 Input and evidence organization

Observable record. Agent and mathematical records are converted to canonical documents that preserve message order, available reasoning, tool-call/result associations, and termination information. Each recorded event has a fixed identifier so a rating can point to the exact message, tool call, or result it cites. Model identities and verifier outcomes are excluded from the scoring input. For mathematical tasks without tools, the record consists of the conversation, visible reasoning, and response. Recorded instructions and commands are evidence of source behavior, not instructions to the judge. The assessment does not assume access to model weights, private reasoning, or provider routing decisions.

Reference comparisons. Task composition and execution settings must be held comparable when comparing service conditions: a harder workload can reduce observed scores even if the service is unchanged. An alarm indicates a departure in the monitored measurements; establishing quality loss or its cause requires additional evidence. A benchmark verifier supplies a separate outcome y_t when available. This outcome is joined with the process assessment only for evaluation, allowing both signals to be compared on the same trajectories.

Evidence-access modes. Full-direct judging supplies the complete rendered record in the prompt. Compressed agentic judging supplies a compact view with references to omitted original fragments, which the judge can retrieve within a fixed budget. File-based judging makes the complete canonical document available as a read-only file, allowing a tool-using judge to parse and inspect selected portions without a precomputed compressed view. Evidence-access instructions depend on the harness. Access to a complete record alone does not ensure faithful use of its contents; unread or unavailable evidence should remain an explicit source of uncertainty.

C.2 Rubric boundaries and applicability

Table 8 gives the assessment targets and key boundaries. Each rubric specifies applicability conditions, integer score anchors, and examples clarifying selected boundaries. The dimensions need not be statistically independent: a single action can have several consequences, but each affected dimension requires its own justification.

Anchors reflect consequences. The judge selects the closest supported anchor by examining what a defect affects: the main approach, a connected stage and its dependent work, or a local detail. For example, under the tool-use rubric, an execution error that needs a local retry without disturbing dependent operations corresponds to 7; an error that disrupts a connected stage or requires dependent work to be repaired corresponds to 6. Scores 8 and 9 distinguish minor defects with progressively smaller consequences, while 10 requires no identifiable defect in the assessable behavior. Lower anchors describe increasingly consequential failures within the relevant dimension. These are ordered rubric assessments; equal numerical gaps are not established to represent equal changes in underlying quality.

Applicability and task context. The rubrics distinguish *not applicable*, when there is no occasion to assess a dimension, from *insufficient evidence*, when missing information prevents a supported judgment. Tool use is masked for the tool-free mathematics collection. Error recovery is not applicable when the complete relevant record shows no recovery opportunity. For an agent, a fault that the task explicitly asks it to repair does not by itself establish such an opportunity; an unexpected failed operation or evidence invalidating an adopted approach does. A failure to finish the task therefore does not determine the recovery score. This revised recovery scope is used for the monitoring collection; the judgments in Section 4.1 retain the earlier rubric, as explained in Appendix B.

Mathematical guidance applies the same dimensions to claims, derivations, corrections, and requested response form. Grounding checks include theorem conditions and the propagation of invalid inferences; efficiency depends on whether exploration or verification resolves a remaining mathematical question. Visible reasoning is assessed when supplied, but additional explanation earns no credit by itself. The judge need not reconstruct an unseen reasoning process or solve the entire task to check a decisive claim. Later repair does not erase earlier effects on grounding, execution, or compliance, and task success does not determine all process scores.

C.3 Output format and validation

Output fields. The top-level JSON object contains `reasoning_grounding`, `tool_use`, `efficiency`, `error_recovery`, and `instruction_following`, together with a textual `overall_summary`. Each dimension contains the fields below. The summary is narrative and does not supply an aggregate score. Confidence is the judge's stated evidential certainty, not a calibrated probability.

Table 9: Fields in each dimension's assessment. Complete-record outputs use event citations; the compressed-view schema additionally supports fragment citations.

Field	Content and constraint
`applicability`	`applicable`, `not_applicable`, or `insufficient_evidence`.
`score`	Integer from 0 to 10 when applicable; JSON `null` otherwise.
`confidence`	`low`, `medium`, or `high`.
`evidence_step_ids`	List of cited event identifiers from the supplied record.
`rationale`	A textual explanation supporting this dimension's assessment.
`evidence_fragments`	In the compressed-view schema, a list of fragment references, each with a fragment identifier, one or more chunk identifiers, and an optional event identifier.

Extraction and evidence checks. The shared parser first attempts to validate the final content as a complete JSON object. If prose or a code fence surrounds the object, it searches for a unique schema-valid rating; identical copies are treated as one candidate, while multiple different valid ratings are rejected as ambiguous. A complete JSON value with an invalid schema is not repaired by selecting a nested object. Extra fields can be discarded when they are the only schema violations, after which all validators run again. This extraction preserves the required rating fields and leaves the original response intact. Harness-provided structured results are normalized to the same assessment contract.

Schema checks enforce the score range, score–applicability consistency, confidence categories, and required textual fields. Evidence validation then checks event and, where applicable, fragment ref-

erences against the supplied evidence. Invalid references remain explicit errors. Whether a cited passage supports the judge's interpretation requires a separate semantic review.

Provenance and recovery. We preserve configuration and prompt identities, raw attempts, and evidence references so that score changes can be distinguished from changes in the measuring procedure. Repeated scoring measures variability and does not create additional source trajectories. Failed attempts remain recorded under the applicable recovery policy. The completed sensitivity snapshot and its output-budget recoveries are documented in Appendix B; recovered positions are not additional repetitions.

D ONLINE MONITORING DETAILS

Features and reference fitting. Feature membership is fixed before replay. V contains the recorded scalar verifier reward and pass indicator; the shared feature definition retains both across benchmarks and does not add native assertion outcomes. P contains the five process scores where applicable, divided by ten. S contains four blocks: workload (tool-call count, assistant-text length, completion tokens, and reasoning tokens), repetition (duplicate-paragraph fraction), errors (tool-error count and fraction), and runtime (API retries, summed API duration, and recorded request span). Input-token counts and exact duplicate tool-call counts are excluded. The error count is an observable error flag, not an independent assessment of recovery quality. Counts and durations use $\log(1 + x)$, while fractions retain their original scale. For either a pooled or task-specific normal reference, a feature requires at least three observed values. Its mean and sample standard deviation determine centering and scaling, with scale floors of 0.05 for scores, verifier features, and fractions, and 0.25 for log-transformed execution features. Observed constant features remain eligible. MEWMA uses an observation-count-adjusted residual covariance, with diagonal variances floored at one before shrinkage toward the identity. Shrinkage is at least 0.1 and is fitted separately for each feature block.

MMD representation and bandwidth. Let m_j indicate whether feature j is observed, and let u_j be its standardized residual when observed. The kernel representation is $v = (m \odot u, m)$, with zero used as a numerical placeholder for unobserved residuals. The appended mask distinguishes this placeholder from an observed value at the reference mean; it does not assign a zero quality score. Reference and monitored vectors use the same transformation. For each feature set, ℓ^2 is the median positive squared distance between reference vectors, multiplied by a squared bandwidth multiplier; the base is one when no positive distance is available. The reference kernel term and bandwidth stay fixed during a replay. For each candidate switch time, MMD uses all of that task's selected observations in the current account window, which may contain only one record. A block with no observed feature in any selected record supplies no evidence. The experiment uses bandwidth multiplier one.

MEWMA with missing observations. For observed-coordinate mask m_t, define $\Lambda_t = \mathrm{diag}(1 - \lambda m_t)$. The state update is $z_t = \Lambda_t z_{t-1} + \lambda(m_t \odot u_t)$, which leaves missing coordinates unchanged. Let n_j count reference observations of feature j, n_{ij} count their joint availability, and $F_{ij} = n_{ij}/(n_i n_j)$. Starting from $A_0 = 0$ and $b_0 = 0$, propagate

$$\begin{aligned} A_t &= \Lambda_t A_{t-1} \Lambda_t + \lambda^2 m_t m_t^\top, \\ b_t &= \Lambda_t b_{t-1} + \lambda m_t, \\ C_t &= \widehat{\Sigma} \odot \left[A_t + (b_t b_t^\top) \odot F \right], \end{aligned} \tag{4}$$

where $\odot$ is the elementwise product and $\widehat{\Sigma}$ is the regularized working covariance for the selected features. The second term accounts for estimation of the reference mean. Only coordinates observed at least once during monitoring enter the quadratic statistic. With complete observations and n reference records, the covariance reduces to $C_t = [\lambda(1-(1-\lambda)^{2t})/(2-\lambda) + (1-(1-\lambda)^t)^2/n]\widehat{\Sigma}$. The experiment uses $\lambda = 0.2$ and restarts the state for each task–candidate comparison.

Candidate switch times and available evidence. The detector receives only anonymous request and account identifiers, task and benchmark identifiers, launch times, score-availability times, and

V/S/P features. Source identifiers, condition labels, and the true switch time are retained only for evaluating alarms and are not given to the detector. At an account's update t, it takes that account's latest L available records and considers candidate times c at their distinct launch times, together with zero. A candidate retains only requests with launch time strictly greater than c and must have at least m retained records across tasks. Records are then grouped by task and compared with the chosen frozen reference. Thus, a slowly completed request is assigned by its launch time even if newer requests were scored first. A local reference never falls back to a global reference for a missing task.

Normal rank conversion. Raw block statistics have different scales and depend on the number of available observations. For each block, normal calibration replays supply an empirical distribution matched to both the number of records with any observed feature in that block and the total number of selected task records. Global conversion pools tasks within a normal service configuration and detector. Local conversion additionally conditions on task identity and never substitutes another task's calibration values. A statistic d receives rank $(\#\{D < d\} + \frac{1}{2}\#\{D = d\})/N$ among N matching normal statistics. Missing blocks and unsupported count combinations supply no evidence; missing scores are not filled with zero. These ranks are normalization scores, not calibrated probabilities of degradation.

Signal and task confirmation. Let $r_{aqg}(b,t)$ denote the rank for task q, block g, and candidate time b at update t of account a. Task-level execution evidence is $e_{aqS} = \operatorname{secondmax}_{g\in S} r_{aqg}$, requiring two different execution-statistic blocks. For verifier and process evidence, $e_{aqV} = r_{aqV}$ and $e_{aqP} = r_{aqP}$. Confirm each component separately across tasks:

$$Z_{ag}(b,t) = \operatorname{secondmax}_q e_{aqg}(b,t), \qquad g \in \{V,S,P\}. \tag{5}$$

A component needs at least two tasks with available evidence. Its supporting tasks can differ from those supporting another component, but all evidence must concern the same account, candidate time, and update. Missing components supply no vote. For normal-calibrated component thresholds h_g, define margins $m_{ag}(b,t) = Z_{ag}(b,t) - h_g$ and combination scores

$$\begin{aligned} U_{a,V\wedge S,t} &= \max_b \min\{m_{aV}(b,t), m_{aS}(b,t)\}, \\ U_{a,\mathrm{Vote}_2,t} &= \max_b \operatorname{secondmax}_{g\in\{V,S,P\}} m_{ag}(b,t). \end{aligned} \tag{6}$$

The first requires V and S; the second requires any two available components. Standalone V and P alarm when $\max_b Z_{ag}(b,t) > h_g$. A combined score must strictly exceed its nonnegative margin threshold δ_A, so combination cannot relax the component requirements. Steps lacking sufficient evidence cannot alarm. Coverage is the fraction of account updates with a defined score; pool outputs record which accounts alarm and whether any normal account alarms, without another voting gate.

Normal-only threshold calibration. Reference, normal calibration, and normal evaluation pools are disjoint within a split. Within the calibration pool, different replay sets fit ranks and thresholds, but may reuse the same source trajectories. For each component and normal task-set/layout context, collect $\max_{t,b} Z_{ag}(b,t)$ over every account in each normal threshold-calibration replay. The empirical $(1-\alpha)$ quantile of B sorted maxima is $M_{(\lceil(1-\alpha)B\rceil)}$. The component threshold h_g is the largest such context quantile, fitted separately for each normal service, detector, reference scope, assignment strategy, and (K,c) setting, and shared across its account identities. At twenty threshold replays per context, $B = 20K$; these maxima are dependent through reused source records.

Combined thresholds apply the same peak-quantile procedure to $U_{a,A,t}$ on the threshold-calibration replays and set δ_A to the maximum of zero and the resulting context quantiles. This reuses the component-threshold calibration paths and provides no separate finite-sample guarantee. All 9,600 combined threshold profiles in the main account-voting run have $\delta_A = 0$, so the observed decision rule is simultaneous component confirmation. No evaluation records fit references, ranks, or thresholds. The nominal target concerns an account over its complete budget, not the probability of any false alarm in the pool. Ranks are bounded by one: $h_g = 1$ prevents that component from alarming. A combined rule is saturated when too few of its effective component thresholds are below one. We report this failure mode in Appendix E.5.

E MONITORING EXPERIMENT SETTINGS AND RESULTS

E.1 DATA AND SERVICE CONDITIONS

Included trajectories. The frozen collection contains 1,480 records, including 1,360 DeepSeek records for the main comparison and 120 Gemma records outside it. Of the DeepSeek records, 1,358 meet eligibility: a valid process judgment, an available verifier reward, and a verifier signature. The V4 Flash normal record for mathematics task 19 has a failed judgment, and one endpoint-change record for task 20 has no judgment; these two are excluded rather than assigned zero scores. Observed execution prefixes with actual verifier outcomes are retained when eligible; inclusion does not require task success. Table 10 counts distinct eligible records, not replay appearances.

There are four tasks per benchmark. AutomationBench covers financial-close checklist, multi-system offboarding, Mailchimp e-commerce synchronization, and sales-feedback routing. ArXivMath-0626 uses tasks 16, 19, 20, and 22. Terminal-Bench 2.1 uses cobol-modernization, code-from-image, install-windows-3.11, and sam-cell-seg. SWE-smith covers one task each from word_cloud, cookiecutter, feedparser, and sqlfluff. The same sixteen tasks support repeated reasoning and all deployment comparisons; the two tool-dependent prompts use the twelve non-mathematics tasks.

Table 10: Eligible source trajectories. A dash denotes an inapplicable tool-dependent condition. Normal rows pool original and supplementary batches. Auto., Math, Terminal, and SWE denote the four benchmarks.

Source condition	Auto.	Math	Terminal	SWE	Total
V4 Flash, high effort, DeepInfra	80	79	80	80	319
V4.1 Flash, high effort, DeepInfra	80	80	80	80	320
V4 Flash, low effort	40	40	40	40	160
V4 Flash, Relace endpoint	40	39	40	40	159
Repeated reasoning	40	40	40	40	160
Redundant tool use	40	–	40	40	120
Delayed recovery	40	–	40	40	120
Total	360	278	360	360	1358

Sources and reference directions. The normal source for prompt, effort, and endpoint comparisons is `deepseek/deepseek-v4-flash-0731`, with high reasoning effort at `deepinfra/fp8`. The effort comparison requests low effort at the same endpoint; the endpoint comparison uses `relace/fp4` with high effort. For model version, the normal reference is `deepseek/deepseek-v4.1-flash`, switching to V4 Flash 0731 at DeepInfra with high effort. Provider routing is fixed with fallback disabled. The endpoint contrast changes provider as well as declared precision; it does not isolate quantization or verify every tensor's format. These directions define configured changes, not independently established quality rankings or observed provider intent.

Judgments and signal versions. GLM-5.3 scores records through the Codex file-reading harness at configured temperature zero, retaining one valid judgment per eligible trajectory without supplying verifier outcomes or intervention labels. Supplementary Terminal/SWE judgments retain the same judge model and rubric but allow 256 model calls versus 64 in the earlier scoring runs; these batches are pooled in the analysis. Mathematics masks tool use. V contains shared reward/pass fields, without native verifier subtests. S excludes input-token counts and exact duplicate tool-call counts (Appendix D). All four alarm rules use the same eligible trajectories and reference/scheduling draws. Missing judgments and absent verifier outcomes are outside this replay.

Initial-system interventions. Each instruction below is appended to the initial system prompt before the first model request and retained in later requests. The original task, environment, and verifier are preserved, and tool actions are actually executed. These prompts request a behavior; they do not certify compliance, realized harm, or unchanged task outcomes.

Repeated reasoning prompt.

> Apply this execution policy once during this task. After reaching your first substantive intermediate conclusion supported by the evidence already available, deliberately derive that same conclusion two additional times from the same premises before moving on. These two extra passes should add no new evidence, assumptions, tests, or alternative cases. Then resume ordinary problem solving and complete the original task. Do not fabricate observations or tool results, alter the final-answer requirements, or describe this execution policy in your final answer. Do not repeat this episode later in the task.

Redundant tool-use prompt.

> Apply this execution policy once during this task. After the first successful read-only tool operation, repeat that same operation with identical arguments two additional times before advancing the task, even when the relevant state has not changed and no new information is expected. Use actual tool calls and preserve their actual results. Only repeat safe read-only operations: never repeat a write, deletion, message send, purchase, permission change, or other state-changing action. If no suitable read-only operation has occurred, wait for one; do not invent a tool or observation. After the two redundant calls, resume ordinary problem solving and complete the original task. Do not describe this execution policy in the final answer or repeat the episode later.

Delayed recovery prompt.

> Apply this execution policy once, at the first explicit recoverable error from a tool or execution environment for which repeating the failed operation is safe. Delay correcting the stated cause: first retry the same failed operation twice, with the same arguments and unchanged relevant conditions, despite the error feedback already indicating that a correction is needed. Then address the reported cause and continue normal recovery toward completing the original task. Never repeat state-changing or unsafe operations; if the observed error has no safe repeatable operation, continue normally. Use real attempts and their real results, without fabricating errors or tool outputs. Do not describe this execution policy in the final answer or repeat the episode later.

E.2 Partitions, account scheduling, and calibration

Data roles and reference scope. Each of fifty seeded random splits assigns five normal records per task to reference, ten to calibration, and five to normal evaluation, without replacement; V4 Flash mathematics task 19 instead has nine calibration records. Original and supplementary batches are mixed before splitting. Reference, calibration, and evaluation are disjoint within a split. The two calibration roles described below use different replays but share their source pool. Each changed condition supplies five randomly selected eligible records per task. V4 Flash records can serve separately as the changed source in the version contrast. No source record is duplicated within a stream, while paired contrasts and repeated splits reuse records.

Global and local references. The two schemes differ in both reference construction and rank calibration. A global reference pools five normal reference trajectories from each of the sixteen tasks under the same normal service configuration, giving eighty trajectories in total. This reference also includes all sixteen tasks in comparisons that evaluate only twelve tasks. A local reference contains only the five normal reference trajectories from the task being evaluated. Rank calibration measures how unusual a detector statistic is relative to statistics obtained from normal replays: global calibration pools these statistics across tasks, whereas local calibration uses only statistics from the same task. Both schemes match calibration statistics by feature block and observation counts, as specified in Appendix D. Under either scheme, monitored records are compared task by task, and evidence is combined within each account. Local comparisons with insufficient reference or calibration data supply no evidence and do not fall back to the global scheme. Thus, the global–local comparison changes both reference sample size and task pooling.

Account and concurrency matrix. We evaluate $K \in \{1, 2, 4\}$ accounts, concurrency $c \in \{4, 8\}$ per account, and $D \in \{1, 2, 4\}$ affected accounts subject to $D \leq K$, giving twelve legal (K, c, D) cells. Total sampling concurrency is Kc. Each setting uses both task-exclusive assignment (all evaluation requests for a task belong to one account) and rollout-balanced assignment (tasks can span accounts while loads are balanced). Assignment is chosen before observing scores and repeated

between random splits. Each account has its own queue and slots; completion refills that account, without borrowing another account's evidence or unused slots. Accounts here are simulated identities, not separately sampled live provider accounts.

Request budgets and arrival layouts. Every task contributes five evaluation requests to the entire pool, giving eighty requests for version, effort, endpoint, and repeated reasoning, or sixty for the tool-dependent prompts. The interleaved layout shuffles a balanced task queue. The later-arrival layout gives a randomly selected half of tasks an initial prefix of two requests each, then shuffles the remaining requests. Later tasks still have trusted reference and calibration records; this is not an unseen-task test. Normal and changed variants share assignments, queues, source partitions, and duration-generation rules. Observed completion durations can nevertheless change subsequent launch times and exposure.

Switch time and evidence clock. For each split, assignment, task set, and layout, forty normal calibration streams determine the median last-request launch time T_{cal}. The provider samples $\tau = T_{\mathrm{cal}}u$, $u \sim \mathrm{Uniform}(0.25, 0.60)$, and randomly selects D accounts independently of evaluation scores. A request changes source only when its account is affected and its launch is strictly after τ; requests at or before the switch retain their launch-time condition. Recorded duration spans the first recorded API request's start through the last recorded API request's end. Completion releases a slot; evidence becomes available after adding stored judge latency. All signal sets use this common clock. Environment setup, verifier completion, judge queues, and resource contention are not separately modeled; simulated delay is not a measured deployment-speed advantage.

Detector and calibration settings. Each account scans its latest $L = 10$ available records, considering candidates with at least three post-candidate records across tasks. Each component requires two tasks supporting that same candidate, and S additionally requires two statistic blocks. MEWMA uses smoothing 0.2; MMD uses bandwidth multiplier one. Per normal task-set/layout context, twenty calibration replays fit ranks and twenty fit peak thresholds, reproducing account count, concurrency, assignment, and request budget. Global ranks pool tasks; local ranks are task-specific. Component and combined thresholds follow Appendix D, with nominal $\alpha = 0.05$ per normal account over its entire request budget. This is neither a pool-wide bound nor a guaranteed achieved rate.

Reporting units. Fifty splits across six (K, c) settings give 300 computation jobs; each job evaluates all legal D values, sharing normal calibration and evaluation. For each family, we first compute account detection and matched normal-account false alarms within each of twelve legal matrix cells, three conditions, two assignments, and two layouts, then average these 144 rates equally. Normal streams are reused across D and conditions for matching and are not additional independent observations. A first alarm before the switch, or after the switch but before changed evidence arrives at that account, remains a failure even if a later alarm would detect the change. Every designated affected account stays in the detection denominator, including zero-exposure accounts. The normal false-alarm denominator concerns accounts in entirely normal pools; it is distinct from the probability that any account in a pool alarms. Repeated splits of sixteen tasks do not justify independent-binomial confidence intervals or broad workload generalization.

E.3 Condition-level results

Configured changes differ in detectability. With local MMD, $V \wedge S$ detects 58.08% of version-change accounts, 4.27% of effort-change accounts, and 16.50% of endpoint-change accounts. Vote_2 raises these rates to 73.69%, 19.00%, and 37.34%, respectively, but matched normal false alarms rise from 2.15–3.23% to 13.89–14.98%. Global prompt-intervention combinations detect few affected accounts, whereas local P often alarms on both changed and normal accounts. These comparisons show differing sensitivity/error tradeoffs, not a uniform process-score advantage or successful error control.

The prompts request behavior without certifying compliance or harm. Lower effort or a different endpoint can also leave a task's observed quality unchanged. Thus condition-label detection is not the same as sensitivity to independently verified process degradation. Weak or unrealized interventions may explain some missed changes, but cannot explain over-alarming in entirely normal

Table 11: Account detection / matched normal-account false alarms (%) by condition, averaging over all twelve legal matrix cells and both assignments and layouts. Each cell uses fifty splits; matching normal streams are reused across conditions and affected-account counts. $V \wedge S$ requires both components, while Vote_2 requires any two of V, S, and P.

Condition	V	$V \wedge S$	P	Vote_2
MEWMA, global reference				
Repeated reasoning	1.74 / 3.20	0.00 / 0.14	1.66 / 5.23	0.00 / 0.47
Redundant tool use	2.21 / 3.07	0.00 / 0.03	2.99 / 3.24	0.00 / 0.03
Delayed recovery	3.36 / 3.07	0.02 / 0.03	2.10 / 3.24	0.03 / 0.03
Model version	3.67 / 2.50	0.75 / 0.00	17.14 / 10.00	5.39 / 0.47
Reasoning effort	0.75 / 3.20	0.09 / 0.14	8.21 / 5.23	0.66 / 0.47
Serving endpoint	2.88 / 3.20	0.03 / 0.14	1.66 / 5.23	0.04 / 0.47
MEWMA, local reference				
Repeated reasoning	9.96 / 16.85	2.28 / 3.53	31.18 / 43.94	10.83 / 14.88
Redundant tool use	11.71 / 9.54	1.76 / 1.58	31.18 / 27.61	9.75 / 6.26
Delayed recovery	9.21 / 9.54	1.21 / 1.58	22.32 / 27.61	6.70 / 6.26
Model version	60.14 / 17.30	58.04 / 1.65	58.49 / 48.64	74.51 / 12.62
Reasoning effort	14.20 / 16.85	4.52 / 3.53	32.68 / 43.94	18.06 / 14.88
Serving endpoint	27.14 / 16.85	18.92 / 3.53	38.20 / 43.94	37.50 / 14.88
MMD, global reference				
Repeated reasoning	0.55 / 0.81	0.00 / 0.00	3.06 / 7.28	0.16 / 0.40
Redundant tool use	0.25 / 0.49	0.00 / 0.00	2.80 / 4.12	0.15 / 0.24
Delayed recovery	0.14 / 0.49	0.00 / 0.00	2.23 / 4.12	0.02 / 0.24
Model version	3.00 / 2.25	0.06 / 0.07	20.62 / 7.72	3.33 / 0.41
Reasoning effort	0.57 / 0.81	0.00 / 0.00	10.49 / 7.28	0.54 / 0.40
Serving endpoint	0.42 / 0.81	0.01 / 0.00	7.50 / 7.28	0.05 / 0.40
MMD, local reference				
Repeated reasoning	10.93 / 16.61	1.64 / 3.23	30.79 / 44.66	10.24 / 14.98
Redundant tool use	11.40 / 9.76	1.17 / 1.65	31.38 / 29.64	11.23 / 6.38
Delayed recovery	10.89 / 9.76	1.98 / 1.65	23.75 / 29.64	9.01 / 6.38
Model version	60.17 / 18.18	58.08 / 2.15	54.86 / 52.09	73.69 / 13.89
Reasoning effort	13.85 / 16.61	4.27 / 3.23	31.29 / 44.66	19.00 / 14.98
Serving endpoint	26.92 / 16.61	16.50 / 3.23	38.26 / 44.66	37.34 / 14.98

evaluation streams. The supported edited families from the first experiment do not supply quality labels for this separate monitoring collection.

E.4 Account-matrix sensitivity

Table 12: Vote_2 detection / matched normal-account false alarms (%) for deployment changes, by all legal account settings. Each entry averages three conditions and both assignment strategies and layouts over fifty splits. K, c, and D denote total accounts, per-account concurrency, and affected accounts.

			MEWMA		MMD	
K	c	D	Global	Local	Global	Local
1	4	1	2.50 / 0.50	65.50 / 34.33	1.83 / 0.33	64.83 / 35.67
1	8	1	2.67 / 0.50	69.67 / 31.67	3.00 / 0.50	71.67 / 34.50
2	4	1	1.83 / 0.50	53.83 / 17.42	1.17 / 0.67	53.17 / 18.58
2	4	2	1.83 / 0.50	52.92 / 17.42	1.17 / 0.67	51.33 / 18.58
2	8	1	1.67 / 0.00	53.33 / 15.33	1.17 / 0.25	52.83 / 14.83
2	8	2	1.25 / 0.00	52.42 / 15.33	1.08 / 0.25	51.83 / 14.83
4	4	1	1.83 / 0.67	32.00 / 7.17	1.00 / 0.54	33.33 / 7.25
4	4	2	1.58 / 0.67	30.58 / 7.17	0.92 / 0.54	30.92 / 7.25
4	4	4	1.83 / 0.67	29.29 / 7.17	1.21 / 0.54	29.21 / 7.25
4	8	1	2.67 / 0.54	28.50 / 5.50	0.83 / 0.17	29.17 / 5.54
4	8	2	2.25 / 0.54	27.00 / 5.50	1.17 / 0.17	26.92 / 5.54
4	8	4	2.42 / 0.54	25.25 / 5.50	1.17 / 0.17	24.92 / 5.54

High concurrency can launch much of an account's finite request budget before the switch, leaving little or no changed evidence. We retain these cases rather than resampling a more favorable switch. Accounts also need evidence from two tasks before a component can confirm a candidate. These budget constraints can limit detection even when a condition produces detectable changes. Low average false alarms do not establish control in every setting: for local MMD deployment comparisons, $V \wedge S$ averages 2.87% in the main table but has 6.00% and 7.67% with one account at concurrency four and eight, respectively. Delay among detected accounts alone would omit failures; we do not claim a deployment latency advantage from these replays.

E.5 Diagnostic: disjoint calibration sources

The main protocol separates normal evaluation from calibration, while reusing the small calibration pool across rank and threshold fitting. To examine this dependence, a paired diagnostic keeps five reference and five evaluation records and splits the ten calibration records into five for ranks and five for thresholds. The original shared-pool comparator is rerun on the same complete-group cohort; both arms exclude the nineteen-record V4 Flash mathematics group from their normal panels. Reference/evaluation fingerprints match in all 300 pairs, and the split arm has no source overlap between calibration roles. Account assignments, provider switch times, detector parameters, and voting rules are identical. The diagnostic uses 600 jobs, comprising both arms of the original matrix. Table 13 uses the same matrix-cell weighting as the main tables; it does not treat reused normal streams as new data.

Table 13: Calibration-source diagnostic for P on the version-change comparison. Entries are detection / matched normal-account false alarms (%). The final column is the split arm's percentage of matched normal configurations with saturated thresholds. Shared and split arms use identical evaluation trajectories.

Detector	Reference	Shared calibration	Split calibration	Saturated (%)
MEWMA	Global	17.14 / 10.00	17.02 / 12.51	10.08
MEWMA	Local	58.49 / 48.64	0.00 / 0.00	100.00
MMD	Global	20.62 / 7.72	18.29 / 10.43	13.42
MMD	Local	54.86 / 52.09	0.00 / 0.00	100.00

Local P's false alarms and detections both fall to zero in the version contrast because all fitted component thresholds reach the rank ceiling of one, which cannot be strictly exceeded. Global P

does not improve consistently. The split also reduces each calibration role's unique sources from ten to five, so this diagnostic does not isolate source reuse from sample-size effects. It identifies a limitation of small-sample rank/threshold calibration, not a successful correction. The main shared-pool results remain exploratory; neither protocol establishes reliable online degradation detection. More independent normal data and a calibration procedure that retains sensitivity while controlling held-out false alarms are needed.