

Calibrated Triage of AI Agent Trajectories with a Guaranteed Miss Rate

Thiago Caserta

Independent researcher

thiago.caserta@gmail.com

27 October 2026

Abstract

Teams that run language-model agents in production spend a large and unmeasured share of their budget deciding which runs deserve attention. Every run is scored by a second model, a fixed sample goes to human reviewers, and runaway runs are cut by an iteration cap. None of these decisions says anything about the runs that were not examined. We ask whether a monitor with no language model inside, fitted only to a process’s own unlabelled event history, can choose which runs to send to expensive review and attach a statement to the rest: under this policy, the expected share of failures among the unreviewed runs is at most α . The statement follows from conformal risk control on a monotone anomaly score; the contribution is the instrument and the evidence about when it works. We report a pre-registered synthetic benchmark and three real-data studies. On 15,436 public τ^2 -bench trajectories from fourteen agents, the guarantee holds in expectation once calibration failures are out of sample and come from the same agent; the generic structural score, however, barely separates failure from success (AUROC 0.48–0.77 across nine agent–domain pairs), while a task-conditional reference does (0.67–0.90), and a one-line conformance rule matches it. At $\alpha = 0.1$ this sends 36–80% of successful telecom runs to review instead of all of them. On 1,667 and 1,998 OpenHands trajectories on SWE-bench Verified, stopping a coding agent on a calibrated structural alarm is no better than a fixed cap: the expensive run is simply the long run, and nothing in its early structure gives it away. We record three protocol mistakes we made along the way and what each one teaches about certificates of this kind. Code, data adapters and every number in this paper are reproducible from hashed artefacts.

1 Introduction

Agents built on language models produce trajectories: a sequence of model turns, tool calls and tool results that ends in a submitted patch, a booked flight or a handed-off conversation. The cost of a trajectory is dominated by the model calls, and the cost of *knowing whether it went well* is dominated by whatever reads it afterwards: a second model acting as judge, a human reviewer, a regression suite. Bai et al. [1] measured the first cost on SWE-bench Verified: coding agents consume roughly a thousand times the tokens of code chat, runs of the same task differ by up to thirty times, and accuracy peaks at intermediate spend before it saturates. The second cost is rarely measured at all, because the usual policies make it invisible. Judging every run has a cost proportional to traffic and says nothing about judge error. Sampling five percent of runs for human review says nothing about the other ninety-five. A fixed iteration cap stops the longest runs regardless of whether they were about to succeed.

What these policies lack is a statement about the runs that were *not* examined. An auditor, a regulator or a cautious engineer wants to hear: “of the runs nobody looked at, at most this fraction were failures, and here is why that number can be trusted.” Statistical practice has such statements elsewhere. In legal document

review, a classifier’s output is validated by sampling the discarded set and estimating recall with a confidence interval, and courts accept the estimate as a basis for not reading the rest [2]. We wanted the analogue for agent runs, prospective rather than retrospective, and cheap enough to run on every trajectory.

This paper studies one instrument for that purpose. It is deliberately not a language model. It fits a small set of reference operators to a process’s own unlabelled event history: which event follows which, which events must precede which, how long each transition takes, and summary statistics of the session’s recurrence structure and transition graph. A new session is turned into a vector of deviations from that reference, the deviations are standardised and combined into an energy, and the energy is converted into a conformal p-value with finite-sample validity. On top of the resulting score, conformal risk control [3] gives the review policy its guarantee: given n known failures that are exchangeable with future ones, the rule “review every session whose score is at least the k -th smallest failure score, $k = \lfloor \alpha(n + 1) \rfloor$ ” misses a future failure with probability at most α . The guarantee needs no model of the normal class and no assumption about the score. The quality of the score decides only how many *normal* sessions have to be reviewed to honour it, which is where a better score pays.

The question we set out to answer is empirical: in real agent traffic, how much does trajectory structure see of failure, and how far does the guarantee carry? We answer it with one synthetic benchmark whose criteria were written before the final run, and three studies on real data whose hypotheses, criteria and data sources were fixed before the data were opened. Where a round of a study failed for a reason we later understood, we kept the round and the reason in the paper, because the failures are informative about certificates of this kind in general and not only about ours.

Contributions.

1. A non-generative monitor for agent trajectories that produces, for each session, a standardised deviation vector, a conformal p-value per feature group, a combined p-value, an online alarm with a calibrated false-alarm rate, and a replayable provenance record (Section 3).
2. A review-triage policy with a finite-sample bound on the expected miss rate, stated as a certificate over the unreviewed mass, with the two conditions that make it valid in practice: calibration failures must be out of sample with respect to every fitted component, and they must come from the same process as the failures to be caught (Section 3.5, Section 5.2).
3. A task-conditional reference that measures a run against the other runs of the same task rather than against the whole process, and the finding that in task-oriented agents it is this conditioning, not the generic structure, that sees failure (Section 5.3).
4. A pre-registered negative result: on coding agents, a calibrated structural alarm does not stop expensive runs better than a fixed cap, and we explain why (Section 6).
5. A deployment protocol with pre-registered criteria for a production clinical assistant, reported here as protocol only, since the volume to evaluate it does not yet exist (Section 7).

What we do not claim. The monitor does not understand what an agent says or does. It cannot see a failure whose event sequence is normal, and we measure how often that happens (the “blind fraction”). The guarantee is in expectation over the calibration draw and holds under exchangeability; we show one way it breaks (cross-agent calibration) and name another (drift). Nothing here generalises across agents by design: the reference is the process’s own history.

2 Related work

Sequence anomaly detection and process conformance. Our operators are the sequential analogue of conformance checking in process mining [4]. The transition operator is a smoothed bigram model [5]; the

precedence operator is a non-local “ a before b ” constraint in the spirit of declarative process models [6]; recurrence quantification [7] and graph-spectral summaries [8] describe global shape. Conformance checking against a per-case model is the closest ancestor of our task-conditional reference. Isolation forests [9] and nearest-neighbour distances are the generic baselines we compare against.

Conformal inference. Split-conformal p-values are valid in finite samples under exchangeability [10]; Bates et al. [11] analyse their use for outlier detection, including the dependence between p-values that share a calibration set. Conformal risk control [3] extends the coverage guarantee to monotone risks; our miss-rate bound is the special case where the loss is the indicator of a missed positive. Simes’ inequality [12] combines per-group p-values; it controls the level under positive regression dependence [13], which is plausible for p-values computed from one calibration set but not guaranteed, so we check the empirical false-alarm rate. Prediction-powered inference [14] and recent work on statistical guarantees for language-model judges [15] certify an *estimate* of a system’s quality; we certify a *decision* about which runs to examine. The two are complementary, and we return to the combination in the discussion.

Evaluating and economising agents. τ -bench and τ^2 -bench [16, 17] evaluate tool-calling agents on policy-constrained tasks with a database-state reward; SWE-bench Verified [18] evaluates coding agents on resolved issues. LLM-as-judge [19] is the dominant evaluation method, and its cost at scale is one of the motivations for gating it. Bai et al. [1] study where tokens go; AgentDiet [20] shortens trajectories by removing redundant context. We study the orthogonal question of which runs deserve the spend at all. Jev [21] argues for non-generative, typed decision models inside software; we share the premise that generation is something one decides *about*, not *with*.

3 Method

3.1 Events, sessions and symbols

A session is an ordered sequence of events $e = (t, \text{actor}, \text{type}, \text{name}, \text{tokens}, \text{attrs})$ with optional text. Actors are user, agent, tool, system and human; types are message, tool call, tool result, handoff and end. Each event has a symbol $\sigma(e) = \text{actor}:\text{type}[:\text{name}]$; a session is bracketed by sentinels BOS and EOS, and symbols unseen in the reference map to UNK. Text is never required: the structural engine reads symbols, times and token counts. Two optional operators read text through deterministic, versioned extractors (marker counts and a hashed bag-of-words residual), and we report them separately.

3.2 Reference operators

The engine is fitted to unlabelled reference sessions, split by a hash of the session identifier into a fit set and a conformal calibration set (30%). Four operators are learned from the fit set.

Transition. A bigram model $P(\sigma_{t+1} \mid \sigma_t)$ with Dirichlet smoothing towards the unigram distribution; each event has a surprise $-\log P$.

Precedence. For each ordered pair (a, b) the empirical confidence that a occurs before b in sessions containing b . Pairs with support at least 20 and confidence at least 0.9 become rules; a session that contains b without an earlier a incurs the surprise $-\log(1 - \hat{c})$. A first-order chain cannot express “ b requires an earlier a anywhere in the past”; the precedence operator exists to cover that gap.

Timing. The log inter-event gap per transition, summarised by a median and a median absolute deviation, with hierarchical fallback from pair to target symbol to global. Deviations are robust z-scores.

Shape. Recurrence rate, determinism and longest diagonal line of the symbol sequence; the number of distinct symbols, algebraic connectivity, self-loop mass and three heat-trace moments of the session’s

transition graph. The reference chain’s spectrum also yields a coarse-graining into metastable phases, used for explanation and for an optional feature group.

Together with log length, duration and tokens, the operators define a feature map $\phi(s) \in \mathbb{R}^{32}$ (Appendix A). Each feature belongs to a group (volume, markov, precedence, timing, rqa, graph, text; and the optional phase, content and task groups) and has a direction: “up” features are anomalous only when large, “both” in either tail.

3.3 Energy, p-values and score

Features are standardised with the reference median and MAD, clipped at 20, and direction-aware: for an “up” feature a negative deviation contributes zero. The energy is the Euclidean norm of the clipped vector,

$$z_j = \text{clip}\left(\frac{\phi_j(s) - \text{med}_j}{\text{MAD}_j}, 0, 20\right), \quad E(s) = \|z\|_2, \quad (1)$$

and each group g has its own energy $E_g(s)$ over its features. The energy is not a probability; it is converted into one by the calibration set. For a group g with calibration energies $\{E_g(c)\}_{c \in C}$,

$$p_g(s) = \frac{1 + \#\{c \in C : E_g(c) \geq E_g(s)\}}{|C| + 1}, \quad (2)$$

which satisfies $\Pr[p_g(s) \leq \alpha] \leq \alpha$ when s is exchangeable with C [10]. The session p-value is the Simes combination of the group p-values and the global one, $p = \min_i m p_{(i)}/i$, so that a session is flagged at level α if any group would flag it at the family-wise level. The triage score is $-\log p$ plus a continuous tie-break $10^{-6}E(s)$: conformal p-values are discrete, many normal sessions sit at $p = 1$, and without the tie-break a review threshold can land inside a mass of tied scores and force the policy to review everything (we hit this in Study A). The explanation attached to a measurement lists the features with the largest z , the group they belong to, and the event indices that produced them.

3.4 Streaming monitor

For a prefix of at least three events, an online energy is computed from the features that are meaningful on an incomplete session (surprise maxima, unknown symbols, precedence violations so far, timing maxima, recurrence on the prefix, self-loops, token volume). The alarm threshold is the $(1 - \alpha)$ conformal quantile of the *maximum* online energy over calibration sessions, so that the probability of any alarm during a normal session is at most α . This is the quantity we test as C16 in Study B.

3.5 Triage with a guaranteed miss rate

Let $s_{(1)} \leq \dots \leq s_{(n)}$ be the scores of n sessions known to be of interest (failures, enforcement cases, reviewer-labelled positives), and let $\alpha \in (0, 1)$ be the miss rate one is willing to accept.

Proposition 1. *Set $k = \lfloor \alpha(n+1) \rfloor$ and $\tau = s_{(k)}$ ($\tau = -\infty$ if $k = 0$). If a future positive s^* is exchangeable with the n calibration positives, then the policy “review every session with score $\geq \tau$ ” satisfies $\Pr[s^* < \tau] \leq \alpha$.*

Proof. Under exchangeability the rank of s^* among the $n+1$ scores is uniform on $\{1, \dots, n+1\}$ (ties broken at random), so $\Pr[s^* < s_{(k)}] \leq k/(n+1) \leq \alpha$. $\square$

This is conformal risk control [3] with the miss indicator as the loss. Three remarks matter in practice. First, the bound holds in expectation over the draw of the calibration set; a given calibration set can be unlucky, and with few positives the realised miss rate has a standard error that we report. Second, few positives force a

$n = 40$ known positives (out of sample, same process), $\alpha = 0.1$; the 4 lowest-scoring positives sit below τ



Figure 1: The review policy and its certificate. Ticks are the scores of known positives; the k lowest sit below the threshold. Everything at or above τ is reviewed; for the rest, the expected share of positives is bounded by α . The quality of the score decides how many *normal* sessions land above τ .

conservative threshold: $k = 0$ means review everything, which is the correct answer when nothing is known. Third, the policy is stored with the hash of the positives, the model hash, α and τ , so the set of unreviewed sessions carries a checkable statement: *under policy P , the expected share of positives in this unreviewed mass is at most α* (Figure 1). We call that statement a certificate. Its validity rests on two conditions that we initially got wrong and now treat as part of the method: the calibration positives must be scored out of sample with respect to every fitted component, and they must come from the same process as the positives to be caught (Section 5.2).

3.6 Task-conditional reference

In task-oriented agents the normal run depends on what was asked. A flight cancellation and a seat change call different tools in different orders, and a reference pooled over all tasks averages this away. When the engine is configured with a conditioning key (a task identifier, an intent label, a route), it keeps, per key value, the set of tools called and the set of tool bigrams of each reference run, and measures a new run against the *other* runs of the same key: the number of tools used by at least half of them but absent here (`task_missing`), the number of tools used here but by fewer than half of them (`task_unexpected`), and one minus the largest Jaccard similarity between this run’s bigram set and any other run’s (`task_seqdev`). Reference runs are scored leave-one-out so that fit-set features are not optimistic. Keys with fewer than three other runs yield neutral features. This is the only place where the engine reads session metadata, and it reads exactly one declared key. The three features form the “task” group and enter the energy and the Simes combination like any other group, or can be used as the triage score on their own.

3.7 Provenance

Every measurement carries the engine, marker, content and task operator versions, the model hash (a SHA-256 of the fitted state, invariant to the order of the reference sessions), the configuration hash, and hashes of the input, the intermediates and the output. Measurements are appended to a hash-chained ledger. Replaying a session against a reloaded model reproduces every hash; we verify this in the benchmark. The engine runs in 0.36 ms per session on a laptop CPU.

Table 1: Main comparison on the synthetic benchmark at $\alpha = 0.05$ (3000 reference, 2000 test normals, 100 sessions per failure type and severity). AUROC pools the ten structural types; the two content-only controls are excluded. Per-type recall is in Appendix C.

method	AUROC, easy	AUROC, hard	FPR on normals
Structural engine (all groups)	1.000	0.931	0.042
Structural engine + content operator	1.000	0.989	0.036
B3 k NN, generic features	0.932	0.864	0.050
B3 k NN + bag of words	0.828	0.794	0.056
B1 Markov bigram	0.841	0.828	0.046
B2 Isolation Forest, generic features	0.736	0.669	0.050
B0 session length	0.705	0.638	0.021

4 Pre-registered synthetic benchmark

Before touching real data we wrote a simulator of a customer-support agent (identity check, order lookup, diagnostics, refund, policy check, escalation, with seven metastable phases in the fitted chain) and twelve injected failure types at two severities. Ten are structural: tool loops, skipped verification, order violations, unknown tools, premature closing, latency spikes, PII leaks, toxic replies, clarification loops and early escalation. Two are negative controls whose event sequence is identical to a normal session and only the wording differs (a semantic error and an unsolicited upsell); a structural engine is expected *not* to detect them, and we never pool them into a structural score. Thirteen criteria were fixed before the final run; the run reported here (results hash 4bfa7f6978e5afe3) differs from the pre-registered one only by the score tie-break of Section 3.3, which changes no p-value and no flag. The simulator was written by the same person who wrote the detector, so these numbers are a capability check, not evidence about the world. Baselines use the same conformal threshold on their own score: session length (B0), a Markov bigram surprise (B1), an isolation forest (B2) and a k -nearest-neighbour distance (B3) on generic sequence features, the last also with a bag of words.

Table 1 gives the main comparison. The engine reaches 0.931 AUROC on the hard pool with a false-alarm rate of 0.042 at a target of 0.05; the nearest baseline is the k NN at 0.864. Removing the transition group costs the most (0.875), then timing (0.883) and text markers (0.885). The negative controls behave as designed: recall of the semantic error is at chance for every structure-only method, and the content operator lifts it to 0.65 without changing the structural numbers. The streaming monitor stops injected loops and unknown tools in every case with 34–69% of their tokens unspent, at a false-alarm rate of 0.050 on normal sessions. Read as a gate in front of a judge, at $\alpha = 0.05$ it sends 7.5% of sessions to the judge, avoids 92.5% of judge tokens, and keeps structural recall at 0.79; semantic recall is 0.11, by construction. Version comparison detected three injected parameter changes out of three with a null rejection rate of 0.10 at target 0.05 and localised one of them, which is the one criterion of thirteen that failed. Model and output hashes are invariant to input order, and replay after save and load reproduced every hash.

Figure 2 and Table 2 show the triage policy. With 200 out-of-sample positives, the realised miss rates track α (0.044 at 0.05, 0.107 at 0.1) for every method, as the proposition says they must; the methods differ only in the price. At $\alpha = 0.05$ the engine reviews 31.4% of normal sessions, the k NN 42.0%, and the length and Markov baselines all of them, because their scores cannot order the calibration positives at all. With the content operator the price drops to 2.8%. The number of calibration positives matters: with 20 positives the engine’s miss rate at $\alpha = 0.05$ is 0.059 but the review share is 39.0%, since a small n forces $k = 1$ and a low threshold.

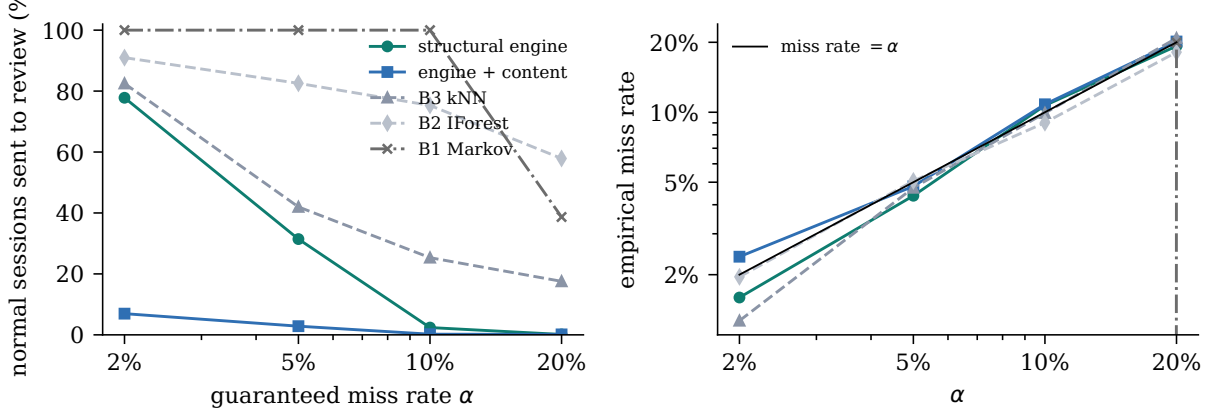


Figure 2: Triage under conformal risk control on the synthetic benchmark, $n = 200$ calibration positives drawn from fresh seeds, mean of 20 draws. Left: share of normal sessions that must be reviewed to honour α . Right: realised miss rate against the bound.

5 Study A: triage on public trajectories of fourteen task-oriented agents

5.1 Data and protocol

τ^2 -bench [17] evaluates agents on airline, retail and telecom tasks against a simulated user and a policy document, and scores each trajectory by the final database state and assertions (reward $\in \{0, 1\}$). Sierra publishes the trajectories of submitted agents. We fixed the data before opening them: the aggregated four-trial files of the fourteen text-only submissions available on 27 October 2026 (Claude 3.7 Sonnet, Claude Opus 4.5, Claude Sonnet 4.5, Gemini 3 Flash, Gemini 3 Pro, GLM-5 thinking, GPT-4.1, GPT-4.1 mini, GPT-5, GPT-5.2, GPT-5.2 without reasoning, o4-mini, Qwen3-Max and Qwen3.5), three domains, four trials per task, 15,436 trajectories in all. The adapter maps an assistant turn with tool calls to one `agent:tool_call:name` event per call, a text turn to `agent:message`, the simulated user’s turns to `user:message` or `user:tool_call` (in telecom the user has tools), and a tool message to `tool:tool_result:name` with a suffix when the environment reports an error. Tokens are the assistant’s prompt plus completion; time is the message timestamp. No message text is kept. A trajectory is a failure when reward < 0.5 . The label is used to choose calibration positives and to evaluate, never to fit the reference. Reference vocabularies have 40 (airline), 49 (retail) and 159 (telecom) symbols.

The protocol that produced the numbers below is the fourth round; the first three and what they taught are in Section 5.2. Trajectories are split three ways by a hash of their identifier: 50% form the reference on which the engine and all baselines are fitted, 20% supply the calibration positives, and 30% are the test set. The reference is pooled per domain across all fourteen agents, because the normal run of a task is a property of the task, while calibration and evaluation are per agent and domain: an agent–domain pair enters only if its own calibration split contains at least 25 failures, which leaves 9 pairs (5 retail, 4 telecom, none in airline, where 200 trajectories per agent leave about ten calibration failures) and excludes 33 (Appendix B). We also report the three pooled test sets, where the calibration positives come from all agents. All methods receive a deterministic tie-break of 10^{-9} seeded by the session hash, i.e. randomised conformal inference, because the task-conformance scores are integer counts and ties otherwise swallow the threshold.

We ran two engine configurations: all groups plus the task group, combined by Simes; and the task group alone as the score. The baselines are B0, B2 and B3 as before, and a new strong baseline B4: the sum of the three task-conformance quantities of Section 3.6 used directly as a score, without the engine. If the engine does not beat B4, the honest conclusion is that the rule suffices.

Table 2: Triage on the synthetic benchmark, $n = 200$ calibration positives, 2000 test positives and 2000 test normals, assumed prevalence 2% for the cost column.

method	α	empirical miss rate	share of normals reviewed	share of review cost avoided
Structural engine (all groups)	0.02	0.016	0.778	0.204
	0.05	0.044	0.314	0.658
	0.10	0.107	0.024	0.957
	0.20	0.193	0.001	0.981
Structural engine + content operator	0.02	0.024	0.070	0.910
	0.05	0.048	0.028	0.953
	0.10	0.108	0.002	0.979
	0.20	0.202	0.001	0.981
B3 k NN, generic features	0.02	0.013	0.825	0.176
	0.05	0.047	0.420	0.570
	0.10	0.099	0.253	0.732
	0.20	0.208	0.176	0.809
B2 Isolation Forest, generic features	0.02	0.020	0.910	0.093
	0.05	0.051	0.825	0.174
	0.10	0.090	0.753	0.239
	0.20	0.181	0.579	0.403
B1 Markov bigram	0.02	0.000	1.000	0.000
	0.05	0.000	1.000	0.000
	0.10	0.000	1.000	0.000
	0.20	0.200	0.387	0.599

The pre-registered criteria were: C17, the realised miss rate is at most $\alpha + 0.02$ for every $\alpha \in \{0.05, 0.1, 0.2\}$ in every pair; C18, at $\alpha = 0.1$ the engine reviews at least five percentage points fewer successes than the best baseline (B4 included) in at least half the pairs; C19, AUROC of score against failure at least 0.70 in at least half the pairs; N4, with permuted labels the AUROC is at chance and the miss rate equals the unreviewed share.

5.2 Three protocol errors, recorded

Round 1: a global reference erases the signal. With one reference per agent and domain and all feature groups, the miss-rate bound held in every pair, but the score barely separated failure from success (AUROC 0.45–0.64, one exception at 0.77), and the policy sent nearly all successes to review. Two things were wrong. Most τ^2 failures are semantic: a wrong argument, a policy violated with an ordinary sequence of calls. And the normal run of a task-oriented agent is conditional on the task, which a pooled reference averages away. A post-hoc check made the second point concrete: grouping the 56 runs of each task (fourteen agents, four trials), the rule “tools the majority of the other runs used and this one did not, plus tools this one used and the majority did not” separated failure from success with AUROC 0.70 (airline), 0.73 (retail) and 0.83 (telecom). The task-conditional group of Section 3.6 and the B4 baseline were pre-registered from that observation. Round 1 also exposed the tie mass at $p = 1$ that motivated the score tie-break.

Round 2: in-sample calibration breaks the guarantee. With the task group, the engine’s AUROC rose, B4 was better still, and the miss-rate bound was violated in 19 of 35 pairs. The cause was in the design: calibration positives were taken from the same split on which the engine and baselines were fitted. Scores of fitted sessions are optimistic, the k -th smallest positive score is therefore too low, and the test set loses more than α . The synthetic benchmark had avoided this by drawing positives from fresh seeds; the real-data protocol had not. The three-way split is the fix, and we now consider “out of sample with respect to every

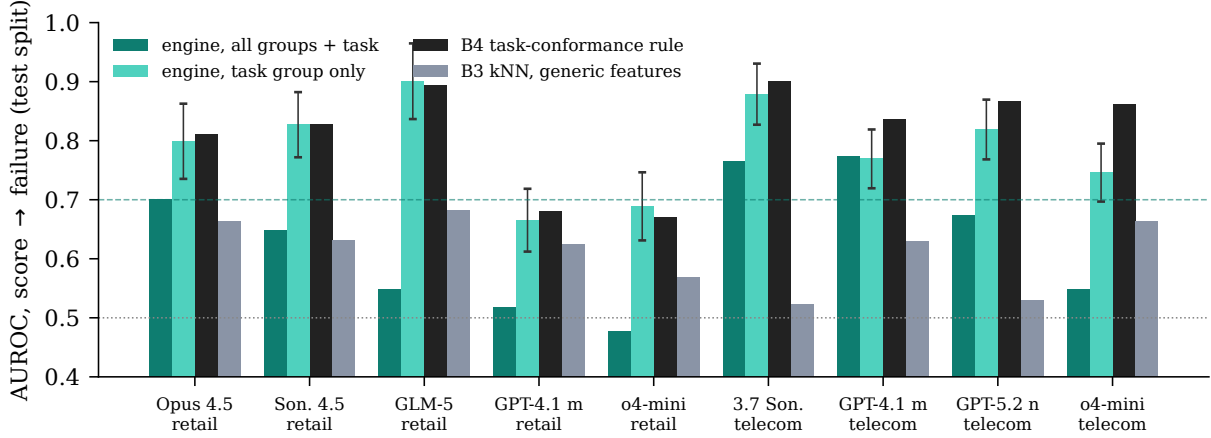


Figure 3: Study A, round 4. AUROC of score against failure on the test split for the nine agent–domain pairs with at least 25 own calibration failures. Error bars: one standard error of the task-only engine’s AUROC. Dashed: the pre-registered 0.70 line.

Table 3: Study A, pooled test sets (calibration positives from all agents). Budget is the share of successes sent to review at $\alpha = 0.1$, with the realised miss rate in parentheses.

domain	reference	calib. failures	test	AUROC, score $\rightarrow$ failure			successes reviewed at $\alpha = 0.1$ (miss rate)		
				engine, all	engine, task	B4 rule	engine, all	engine, task	B4 rule
airline	1412	166	854	0.53	0.67	0.70	0.84 (0.113)	0.77 (0.073)	0.75 (0.081)
retail	3124	310	1890	0.61	0.75	0.76	0.80 (0.112)	0.75 (0.094)	0.84 (0.050)
telecom	3161	248	1895	0.62	0.81	0.83	0.75 (0.095)	0.51 (0.090)	0.30 (0.095)

fitted component” part of the definition of a valid calibration set.

Round 3: failures of another agent are not exchangeable. To rescue pairs with too few own failures, round 3 allowed calibration positives from the pooled failures of all agents. The pairs that violated the bound were almost exactly those pairs: GPT-5.2 on telecom, calibrated on other agents’ failures, missed 38% at $\alpha = 0.1$. The failures of a strong agent are not drawn from the same distribution as the failures of a weak one; the certificate is per process. Round 3 also had a runner bug (a shadowed variable made both engine configurations identical) and is discarded as evidence; its numbers agreed with round 4 where they overlapped.

5.3 Results

The guarantee holds in expectation, per process. On the pooled test sets, with 166 to 310 calibration failures, the realised miss rate at $\alpha = 0.1$ is 0.073 (airline), 0.094 (retail) and 0.090 (telecom) for the task-only engine and 0.081, 0.050 and 0.095 for B4 (Table 3, Figure 4). Across the nine per-pair evaluations, with 25 to 63 calibration failures and 25 to 85 test failures each, the task-only engine’s miss rate at $\alpha = 0.1$ averages 0.070 and ranges from 0.00 to 0.18 (Appendix B). The strict criterion C17 (every α , every pair, within 0.02) passes in 6 of 9 pairs; the excursions are within two standard errors of a bound that, with 25 test failures, has a standard error of about 0.06, and we report them with intervals rather than widen the criterion after the fact. The permutation control N4 passes everywhere.

The generic structure does not see these failures; task conformance does. With all groups combined, the engine’s AUROC ranges from 0.48 to 0.77 and reaches 0.70 in 3 of 9 pairs. With the task group alone it

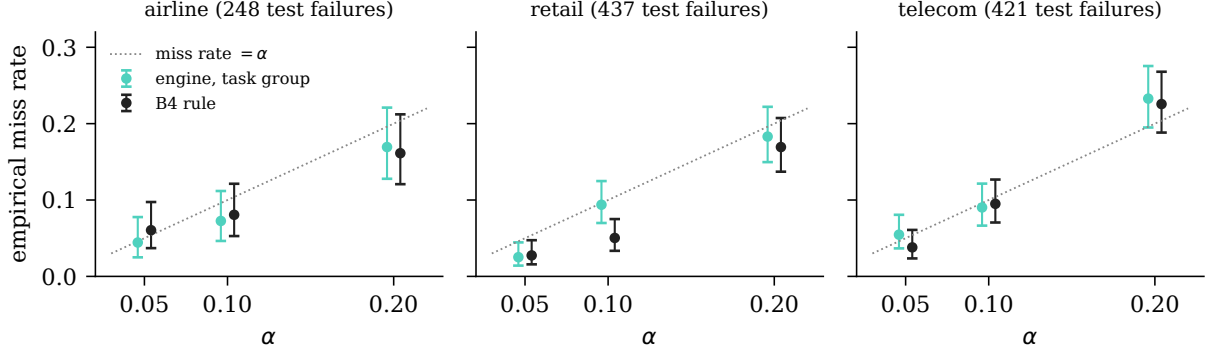


Figure 4: Realised miss rate against α on the pooled test sets, with 95% Wilson intervals, for the task-only engine and the B4 rule.

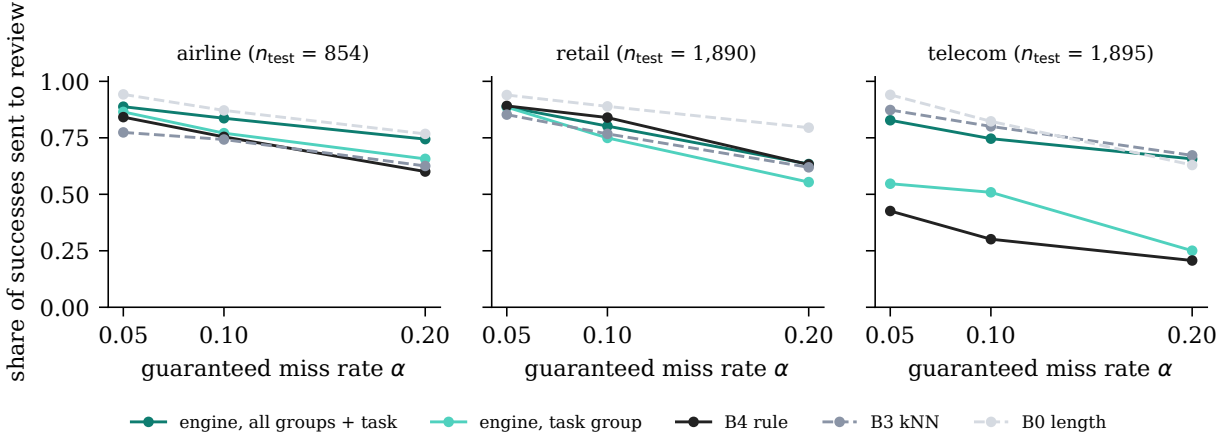


Figure 5: Review budget on the pooled test sets: share of successful runs sent to review as a function of the guaranteed miss rate.

ranges from 0.67 to 0.90 and reaches 0.70 in 7 of 9 (Figure 3). B4, the one-line rule, ranges over the same interval and reaches 0.70 in 7 of 9; the generic k NN never does (0.52–0.68). Combining the task group with seven groups that cannot see the failure makes the score *worse*: Simes’ combination dilutes the one informative p-value with seven uninformative ones. C19 passes for the task-only configuration and fails for the combined one.

What the budget looks like. In telecom, to catch 90% of failures the task-only engine sends 36–80% of successful runs to review and B4 sends 28–66%; the all-groups engine sends 58–81%. On the pooled telecom test set the figures are 51% and 30% of successes, i.e. 60% and 44% of all traffic (Figure 5). Read as a gate in front of a judge, that is 40 to 56% of judge calls avoided at a certified 10% miss rate. In retail the budget is worse (47–97% of successes), and airline cannot be evaluated with honest calibration at this volume. C18 passes in 2 of 9 pairs: the engine does not beat the rule by five points.

The blind fraction. The share of failures that score below the median success, which no threshold on this score can catch without reviewing most successes, is 0.03–0.13 in telecom and 0.08–0.30 in retail for the task-only engine. That is the ceiling of what task-level structure reaches without reading content in these domains.

Reading. The contribution the data support is not the generic structural score. It is the principle that

Table 4: Study B. Prefix energy does not anticipate the expensive run; stopping on the alarm does not beat a fixed cap at a loss anyone would accept; the calibration of the alarm holds where timing is exchangeable.

	Claude Sonnet 4.5	GPT-5
trajectories (reference / test)	1,667 (1,167 / 500)	1,998 (1,385 / 613)
resolved	0.73	0.60
median tokens per trajectory	2.62M	0.78M
tokens, p90 / p10	2.8×	5.2×
same task, max / min: median (p90)	1.4× (2.0×)	2.0× (19.2×)
symbols in the reference vocabulary	234	125
H9: AUROC of prefix energy, $k = 10/20/30$	0.47 / 0.51 / 0.48	0.55 / 0.52 / 0.56
H9: AUROC of tokens spent so far, $k = 10/20/30$	0.64 / 0.56 / 0.54	0.62 / 0.57 / 0.60
H10: success loss at $\alpha = 0.01/0.02/0.05/0.1/0.2$	0.004 / 0.010 / 0.034 / 0.070 / 0.126	0.011 / 0.018 / 0.041 / 0.080 / 0.126
H10: token savings at the same α	0.006 / 0.011 / 0.044 / 0.097 / 0.146	0.014 / 0.025 / 0.058 / 0.128 / 0.189
H10: margin over the better of cap and budget at matched loss, points	+0.6 / +0.8 / +3.4 / +7.9 / +10.2	+1.4 / +0.1 / +0.9 / -0.1 / -0.3
H11: AUROC of final energy $\rightarrow$ unresolved	0.53	0.57
C16: alarm rate within $\alpha \pm 0.02$	5 of 5	3 of 5
alarm rate at $\alpha = 0.1 / 0.2$	0.110 / 0.192	0.139 / 0.232

the reference must be conditional on the task, the calibrated policy with its certificate, the measured review budget as the quantity to optimise, and the provenance around it. Where a one-line rule reaches the same AUROC, the method should use the rule as its score and keep the rest. That is what the engine now does: the triage score is chosen per group, Simes’ combination is kept for the anomaly flag (detect what any group detects) and not for ranking.

6 Study B: calibrated stopping of coding agents (negative)

6.1 Hypotheses, data and criteria

If expensive runs are structurally different runs rather than harder tasks, as the within-task variation reported by Bai et al. [1] suggests, a calibrated online alarm might stop them early and beat a fixed iteration cap on the cost–success frontier. We pre-registered three hypotheses and three criteria before opening the data. H9: the maximum online energy over the first $k \in \{10, 20, 30\}$ events separates runs that end in the top cost decile of their model from the rest, with AUROC at least 0.70 at $k = 20$ and above two baselines (tokens spent so far, tool calls so far) at two of three k (C14). H10: stopping on the first alarm at some α loses at most one point of success rate against never stopping and saves at least ten points more tokens than the best fixed event cap and the best token budget at the same loss (C15). C16: the alarm rate on held-out trajectories lies within $\alpha \pm 0.02$ for every α . H11, reported without a criterion: AUROC of the final energy against the resolved label. N3: permuted labels give AUROC near 0.5.

The data are the OpenHands trajectories on SWE-bench Verified released by Bai et al. (500 tasks, four runs per model, at most 100 iterations, per-call token usage, resolved labels). We converted two models with different log formats: Claude Sonnet 4.5 (1,667 trajectories, 73% resolved, median 2.6M tokens and 137 events) and GPT-5 (1,998 trajectories, 60% resolved, median 0.8M tokens and 68 events). The adapter gives each tool call a sub-symbol (the editor command, or the first word of a shell command), marks tool results with a non-zero exit code or a traceback, drops the task statement and system prompt, which are identical across runs, and uses the real call timestamp as time. The reference is per model (70% by hash, with the engine’s own 30% calibration inside), the test set is the remaining 30%, and the label is used only to evaluate.

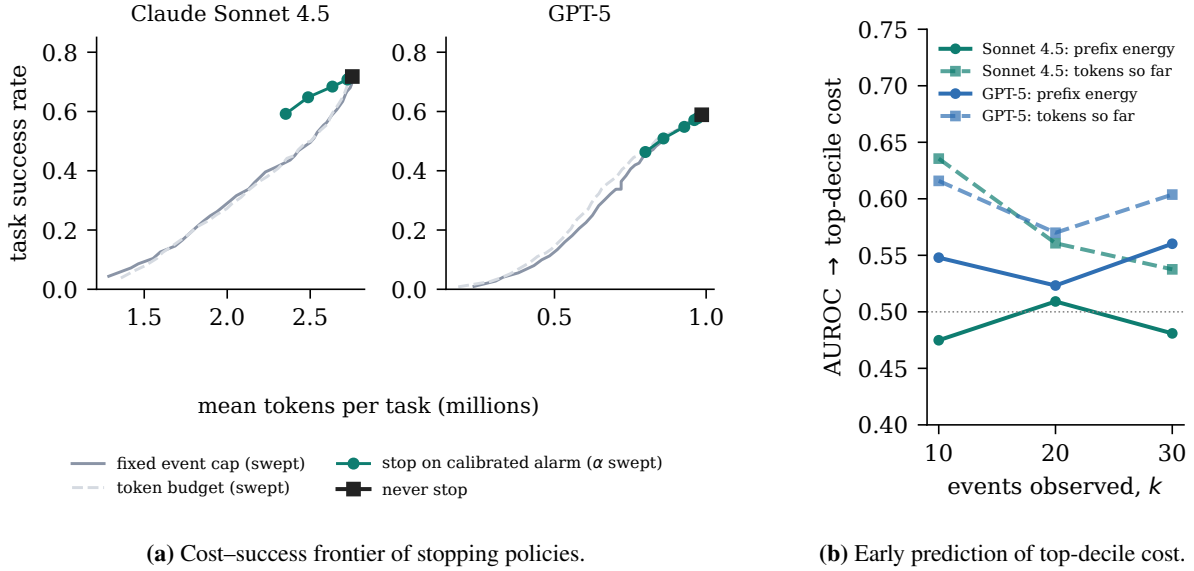


Figure 6: Study B. Left: for each policy, mean tokens per task against success rate as its parameter is swept; the alarm curve is the calibrated monitor at $\alpha \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$. Right: AUROC of the prefix energy and of tokens spent so far after k events.

6.2 Results

Both pre-registered criteria fail on both models (Table 4, Figure 6). The prefix energy is at chance for predicting the expensive run (AUROC 0.51 and 0.52 at $k = 20$); the context already consumed after ten events is a better predictor (0.64 and 0.62), and the number of events predicts the top cost decile with AUROC above 0.96, which is to say that the expensive run is the long run. Stopping on the alarm loses successes at the rate it stops runs: at $\alpha = 0.1$ the success loss is 0.070 and 0.080 for token savings of 0.097 and 0.128. At the one-point loss the criterion allows, the margin over the better of cap and budget is +0.8 and +1.4 points, not ten. On Claude Sonnet 4.5 the alarm does dominate the cap at matched loss as α grows (+10.2 points at $\alpha = 0.2$), but at a success loss of 0.126 that nobody would accept; on GPT-5 the two coincide. The within-task spread is smaller than the headline figure of [1] suggests once one stays inside a model: the median ratio of the most to the least expensive run of the same task is 1.4 and 2.0, with a thin tail (p90 of 19.2 on GPT-5). The final energy separates unresolved from resolved runs only weakly (AUROC 0.53 and 0.57).

What held is the calibration. On Claude Sonnet 4.5 the alarm rate on held-out trajectories is within $\alpha \pm 0.02$ at every α (5/5). On GPT-5 it is within the band at three of five, with 0.139 at $\alpha = 0.1$ and 0.232 at $\alpha = 0.2$; the alarms responsible are dominated by the timing feature, and the trajectories were collected on different days with different API latency, so reference and test are not exchangeable in time. The permutation control passed on Claude and failed on GPT-5 by a margin consistent with the band having been set at one standard error for 61 positives, which was our mistake in pre-registration and is reported as such.

The reading is simple. In these coding agents an expensive run is not an abnormal run that reveals itself early; it is an ordinary run that goes on for longer, and a structural monitor of the kind described here has nothing to see until the cost has been incurred. Post hoc, the energy of the *complete* trajectory separates the runs that hit the 100-iteration cap with AUROC 0.63 and 0.86, on 8 and 5 positives, which is suggestive and nothing more. Calibrated stopping of coding agents is not a capability we claim.

7 Study C: deployment protocol in a production clinical assistant

The third study is a deployment rather than a benchmark, and it is reported here as protocol because the volume needed to evaluate it does not yet exist. The system is a clinical scheduling and patient-communication assistant used by medical clinics in Brazil, with two agents running a tool-calling loop of at most six rounds per turn. Each run writes one usage row (route, escalation level, rounds, stop reason, tools, failures, latency, trace identifier) and the conversation log stores every message and tool event with a timestamp. A nightly judge model scores up to sixty conversations on a five-point scale, and scores of three or less become review cases.

The engine is attached at three points without reading any patient text, which never leaves the clinic’s database: the export carries symbols, times and token counts only. First, a *judge gate*: only conversations whose structural p-value is below the clinic’s α , or the lowest-scoring ones the nightly budget allows, are sent to the judge, and the judge’s scores on the conversations it does see supply the labels that measure the miss rate. Second, a *round brake*: the online monitor evaluates the prefix at each tool result and, on an alarm with calibrated false-alarm rate, interrupts the turn and opens a human handoff before the loop’s full re-execution, its “second chance” and its directed retry, which are where repeated spend occurs. Third, *review triage*: the policy of Section 3.5 selects the conversations the clinical team reads, with the certificate for the rest as the basis for the clinic’s audit record.

Two criteria are fixed. C20, judge gate: with $\alpha = 0.1$ and the judge’s label, the miss rate is at most 0.12 and at least half of the judge calls are avoided, on at least 300 judged conversations. C21, round brake: in replay of runs that escalated or retried, the alarm precedes the re-execution in at least half of the cases, with a false-alarm rate at most $\alpha + 0.02$ on runs that ended in a text reply in the first round. The intent label that the system already computes per turn is the conditioning key of Section 3.6; the deployment will measure what a coarse intent delivers, where τ^2 ’s task identifier measured the ceiling. Two weeks of production produced 149 usage rows, which is not enough to calibrate; per-round token and latency instrumentation has been switched on so that the data accumulate in the right shape, and the study will be reported when the criteria can be evaluated.

8 Discussion

What trajectory structure sees. Across the synthetic benchmark and the two real studies the pattern is consistent. Structure sees failures that *are* structure: loops, skipped steps, unknown tools, order violations, premature endings, and, in task-oriented agents, the absence of a tool the task needs or the presence of one it does not. It does not see a wrong argument, a violated policy with an ordinary call sequence, or a coding run that is simply long. The blind fraction we report is the honest measure of the second class, and in retail it is a fifth to a third of failures. A certificate built on structure alone is a certificate about structural failure, and a deployment that wants more must add a reader of content, a judge or a human, behind the gate; what the gate then certifies is that the reader saw everything the structure could flag at a known miss rate.

The score is not the contribution; the certificate is. We expected the engineered structural score to be the result of this paper. It was not. In the one setting where failure labels exist at scale, a one-line rule computed from the same conditioning reaches the same AUROC, and combining it with the generic groups makes it worse. What survives is the part that does not depend on the score being clever: the conformal calibration, the miss-rate bound, the budget as the measured outcome, and the two conditions on the calibration set. A better score, learned with labels or reading content, slots into the same policy and lowers the budget without touching the guarantee. We think this is the right division of labour and would have designed for it from the start had we known.

Exchangeability and its two failure modes. The guarantee is only as good as the claim that future positives look like calibration positives. We saw it fail two ways. Calibrating on sessions the reference was fitted on produced optimistic scores and a threshold too low (round 2). Calibrating on another agent’s failures produced a threshold for the wrong distribution (round 3). A third way, not tested here, is drift: an agent whose model, prompt or tools change produces new failure shapes that a stale calibration set does not cover. The version-comparison statistic of the engine detects the change; it does not repair the certificate, which has to be re-issued with fresh positives. A certificate should therefore carry its calibration date and the model hash it was issued against, and a verifier should refuse one whose process has since changed.

Relation to validation in document review. The elusion test used to validate technology-assisted review samples the discarded set after classification and estimates the recall achieved [2]. Our certificate is a prospective version of the same promise: the miss rate is fixed as policy before review, the threshold is deterministic given the positives’ hash, and nothing is sampled after the fact. The two are complementary. A small random sample of the unreviewed mass, labelled by a human or a judge and combined with the gate’s own scores through prediction-powered inference [14], would turn the prospective bound into a retrospective estimate with an interval at a fraction of the cost of reading the mass. We have not done this and think it is the natural next step.

Cost. The monitor runs in a fraction of a millisecond per session and needs no accelerator. In the deployments we have in mind the comparison is not with the monitor’s cost but with the judge calls it avoids and the review hours it redirects, and the budget figures of Section 5.3 are the quantities that decide whether it pays.

Limitations. The real-data evidence comes from one benchmark family for task agents and one for coding agents, both with simulated users or fixed tasks; production traffic is less regular and its failure labels are noisier. Nine agent–domain pairs with 25 to 85 test failures each give miss-rate estimates with standard errors of several points, which is why we show intervals rather than pass–fail. The reference is per domain and the task identifier is finer than any intent label a production system has; Study C will measure the gap. The content operators were evaluated on synthetic text only. And the conditions under which Simes’ combination controls the level were checked empirically, not proved for these p-values.

9 Reproducibility

Everything reported here is produced by scripts from raw data. The synthetic benchmark is `bench/run_benchmark.py` with seed offset 30000 (results hash 4bfa7f6978e5afe3, model hash 271b29413ca3f246). Study A is `bench/real/run_triage.py` with the τ^2 adapter `bench/real/tau2.py`; Study B is `bench/real/run_e8.py` with the two OpenHands adapters; the clinical export is a SQL script that carries no message text. Figures, tables and every number in the prose are generated from the result files by `papers/tex/build_assets.py`, so that no value is typed by hand. The τ^2 trajectories come from Sierra’s public bucket; the OpenHands trajectories from the release of Bai et al., which declares no licence, and we redistribute nothing derived from them. Each fitted model is identified by a SHA-256 of its state and each measurement by the hashes of its input, intermediates and output; the ledger of measurements is hash-chained and verifiable. Pre-registration documents, round-by-round, are part of the repository.

Acknowledgements. The τ^2 -bench team at Sierra and Bai et al. made their trajectories public, without which Studies A and B would not exist. The clinical deployment of Study C is run with the clinics’ consent and no patient text leaves their systems.

References

- [1] L. Bai, Z. Huang, X. Wang, J. Sun, R. Mihalcea, E. Brynjolfsson, A. Pentland, J. Pei. How do AI agents spend your money? Analyzing and predicting token consumption in agentic coding tasks. arXiv:2604.22750, 2026.
- [2] M. R. Grossman, G. V. Cormack. Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review. *Richmond Journal of Law and Technology* 17(3), 2011.
- [3] A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, T. Schuster. Conformal risk control. *ICLR*, 2024.
- [4] W. M. P. van der Aalst. *Process Mining: Data Science in Action*. Springer, 2nd edition, 2016.
- [5] S. F. Chen, J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language* 13(4), 1999.
- [6] M. Pesic, H. Schonenberg, W. M. P. van der Aalst. DECLARE: full support for loosely-structured processes. *EDOC*, 2007.
- [7] N. Marwan, M. C. Romano, M. Thiel, J. Kurths. Recurrence plots for the analysis of complex systems. *Physics Reports* 438, 2007.
- [8] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
- [9] F. T. Liu, K. M. Ting, Z.-H. Zhou. Isolation forest. *ICDM*, 2008.
- [10] V. Vovk, A. Gammerman, G. Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.
- [11] S. Bates, E. Candès, L. Lei, Y. Romano, M. Sesia. Testing for outliers with conformal p-values. *Annals of Statistics* 51(1), 2023.
- [12] R. J. Simes. An improved Bonferroni procedure for multiple tests of significance. *Biometrika* 73(3), 1986.
- [13] S. K. Sarkar. Some probability inequalities for ordered MTP_2 random variables: a proof of the Simes conjecture. *Annals of Statistics* 26(2), 1998.
- [14] A. N. Angelopoulos, S. Bates, C. Fannjiang, M. I. Jordan, T. Zrnica. Prediction-powered inference. *Science* 382(6671), 2023.
- [15] Y. Sheng et al. Conformal prediction intervals for LLM-as-a-judge scores. *NeurIPS*, 2025. arXiv:2509.18658.
- [16] S. Yao, N. Shinn, P. Razavi, K. Narasimhan. τ -bench: a benchmark for tool-agent-user interaction in real-world domains. arXiv:2406.12045, 2024.
- [17] V. Barres, H. Dong, S. Ray, X. Si, K. Narasimhan. τ^2 -bench: evaluating conversational agents in a dual-control environment. arXiv:2506.07982, 2025. Public trajectories: `s3://sierra-tau-bench-public`.
- [18] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, K. Narasimhan. SWE-bench: can language models resolve real-world GitHub issues? *ICLR*, 2024. SWE-bench Verified: OpenAI, 2024.
- [19] L. Zheng et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *NeurIPS*, 2023.
- [20] Y.-A. Xiao, P. Gao, C. Peng, Y. Xiong. Reducing cost of LLM agents with trajectory reduction (AgentDiet). *FSE*, 2026. arXiv:2509.23586.
- [21] TypeSafe AI. Introducing System One models and Jev. Company publication, September 2026.

A Feature groups

Table 5: The 32 features. Groups phase, content and task are opt-in. Direction “up” means only a positive deviation contributes to the energy.

feature	group	direction	meaning
len_log	volume	both	log number of events
dur_log	volume	both	log wall-clock duration
tok_log	volume	up	log tokens billed
nll_mean	markov	up	mean transition surprise $-\log P(\sigma_{t+1} \sigma_t)$
nll_max	markov	up	maximum transition surprise
unk_count	markov	up	events with a symbol unseen in the reference
prec_max	precedence	up	largest precedence-violation surprise
prec_count	precedence	up	number of precedence violations
tim_max_z	timing	up	largest robust z of a log inter-event gap
tim_mean_abs_z	timing	up	mean $ z $ of log inter-event gaps
rqa_rr	rqa	up	recurrence rate of the symbol sequence
rqa_det	rqa	up	determinism (share of recurrences on diagonals)
rqa_lmax	rqa	up	longest diagonal line
g_nodes	graph	both	distinct symbols in the session graph
g_lambda2	graph	both	algebraic connectivity of the session graph
g_selfloop	graph	up	self-loop mass
g_heat_0.5	graph	both	heat-trace moment, $t = 0.5$
g_heat_2	graph	both	heat-trace moment, $t = 2$
g_heat_8	graph	both	heat-trace moment, $t = 8$
phase_nll_mean	phase	up	mean surprise under the coarse-grained (metastable phase) chain
phase_entropy	phase	both	entropy of the phase occupation
txt_agent_pii	text	up	deterministic PII pattern count in agent text
txt_agent_toxic	text	up	toxicity marker count in agent text
txt_agent_refusal	text	up	refusal marker count in agent text
txt_user_negative	text	up	negative-sentiment marker count in user text
c_agent_resid	content	up	hashed TF-IDF subspace residual, agent text
c_agent_oov	content	up	out-of-vocabulary share, agent text
c_user_resid	content	up	subspace residual, user text
c_user_oov	content	up	out-of-vocabulary share, user text
task_missing	task	up	tools used by at least half of the other runs of the task but absent here
task_unexpected	task	up	tools used here but by fewer than half of the other runs
task_seqdev	task	up	one minus the best Jaccard similarity of tool bigrams with another run of the task

B Study A: per-pair results and excluded pairs

Table 6: Study A, round 4, per agent–domain pair with at least 25 own calibration failures. “engine, all” is all groups plus task combined by Simes; “engine, task” is the task group alone. The miss-rate interval is a 95% Wilson interval given the number of test failures.

agent	domain	n_{test}	failures	AUROC, score $\rightarrow$ failure				successes reviewed, $\alpha = 0.1$		miss rate, $\alpha = 0.1$	
				engine, all	engine, task	B4 rule	B3 kNN	engine, task	B4 rule	engine, task	95% CI
Claude Opus 4.5	retail	145	25	0.70	0.80	0.81	0.66	0.83	0.91	0.00	[0.00, 0.13]
Claude Sonnet 4.5	retail	145	37	0.65	0.83	0.83	0.63	0.47	0.83	0.11	[0.04, 0.25]
GLM-5 (thinking)	retail	137	25	0.55	0.90	0.89	0.68	0.97	0.97	0.00	[0.00, 0.13]
GPT-4.1 mini	retail	135	44	0.52	0.67	0.68	0.62	0.92	1.00	0.00	[0.00, 0.08]
o4-mini	retail	132	34	0.48	0.69	0.67	0.57	0.78	0.90	0.09	[0.03, 0.23]
Claude 3.7 Sonnet	telecom	126	59	0.76	0.88	0.90	0.52	0.37	0.28	0.07	[0.03, 0.16]
GPT-4.1 mini	telecom	137	76	0.77	0.77	0.84	0.63	0.36	0.33	0.18	[0.11, 0.29]
GPT-5.2 (no reasoning)	telecom	135	56	0.67	0.82	0.87	0.53	0.47	0.35	0.14	[0.07, 0.26]
o4-mini	telecom	144	85	0.55	0.75	0.86	0.66	0.80	0.66	0.04	[0.01, 0.10]

Table 7: Pairs excluded for lack of own calibration failures.

domain	agents excluded, with the number of own calibration failures in parentheses
airline	claude-3-7-sonnet (16), claude-opus-4-5 (9), claude-sonnet-4-5 (8), gemini-3-flash (11), gemini-3-pro (9), glm-5-think (7), gpt-4-1-mini (20), gpt-4-1 (18), gpt-5-2-none (4), gpt-5-2 (3), gpt-5 (20), o4-mini (22), qwen3-max (11), qwen3.5-397b-a17b-think (8)
retail	claude-3-7-sonnet (18), gemini-3-flash (23), gemini-3-pro (13), gpt-4-1 (18), gpt-5-2-none (21), gpt-5-2 (24), gpt-5 (19), qwen3-max (20), qwen3.5-397b-a17b-think (20)
telecom	claude-opus-4-5 (3), claude-sonnet-4-5 (7), gemini-3-flash (7), gemini-3-pro (6), glm-5-think (1), gpt-4-1 (21), gpt-5-2 (12), gpt-5 (1), qwen3-max (3), qwen3.5-397b-a17b-think (2)

C Synthetic benchmark: recall per failure type

Table 8: Recall at $\alpha = 0.05$ per hard failure type. [†] content-only controls, expected at chance for every method that does not read text.

failure type (hard variant)	engine	+content	B1	B3	B2	B0
tool_loop	0.98	0.98	0.79	1.00	0.73	0.49
skip_verification	1.00	1.00	0.97	1.00	0.27	0.09
order_violation	1.00	1.00	1.00	1.00	0.04	0.00
unknown_tool	1.00	1.00	1.00	0.27	0.06	0.10
premature_close	0.92	0.93	0.87	0.98	0.03	0.45
latency_spike	0.97	0.97	0.06	0.08	0.08	0.01
pii_leak	1.00	1.00	0.05	0.05	0.04	0.02
toxic_reply	0.02	1.00	0.03	0.02	0.00	0.01
clarification_loop	1.00	1.00	1.00	1.00	0.54	0.43
early_escalation	0.27	0.30	1.00	0.43	0.00	0.69
semantic_error [†]	0.03	0.65	0.05	0.06	0.10	0.02
unsolicited_upsell [†]	0.06	0.62	0.02	0.03	0.07	0.00

D Pre-registered criteria and outcomes

Table 9: Every criterion fixed before the corresponding data were opened, and its outcome.

study	criterion	statement	outcome
synthetic	C1–C13	detection, false-alarm, streaming, gating, drift, determinism and triage criteria of the benchmark	12 of 13 pass (drift localisation fails)
A	C17	miss rate $\leq \alpha + 0.02$ for every α in every pair	pools: pass; pairs: 6 of 9 (task-only)
A	C18	engine reviews ≥ 5 points fewer successes than the best baseline in half the pairs	2 of 9: fail
A	C19	AUROC ≥ 0.70 in half the pairs	task-only 7 of 9: pass; all groups 3 of 9: fail
A	N4	permuted labels at chance	pass
B	C14	prefix energy AUROC ≥ 0.70 at $k = 20$ and above both baselines	fail, both models
B	C15	alarm stopping: loss ≤ 1 point and ≥ 10 points more savings than cap and budget	fail, both models
B	C16	alarm rate within $\alpha \pm 0.02$	5/5 (Claude), 3/5 (GPT-5)
B	N3	permuted AUROC in $[0.45, 0.55]$	pass (Claude); fail (GPT-5, band too narrow for 61 positives)
C	C20, C21	judge gate and round brake (Section 7)	pending volume