GUARDED COMMITS: TRANSACTIONAL HUMAN APPROVALS FOR LLM WORKFLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) workflows often require human approval before an irreversible action, such as merging a pull request that a coding agent opened. Many agent frameworks record the approval apart from the component that performs the action. Unless the code-hosting platform is configured to enforce it, nothing checks that the approval covers what is merged. We ask whether this matters and whether approval could be automated instead. In 30,796 agent-authored pull requests, 83.4% of merges carry no approving review, and 12.6% of 3,880 approved merges went in at a code version other than the one approved. We then use Learn-then-Test to bound, with stated confidence, the share of auto-approved items that a human rejected. On 8,077 reviewed agent pull requests and 17,007 writing-assistant decisions, no method that sees only the proposed change and its origin certifies any automation at a 5% bound on the full held-out sets at our sample size, and LLM judges certify at most 3.1% on a stricter subset. This holds for trained predictors and for frontier LLM judges, the best of which reaches an area under the ROC curve (AUROC) of 0.77. The reviewers' own pre-decision signals, which partly encode the decision, raise AUROC to 0.88. These results motivate *guarded commits*, which store each approval with its evidence in an append-only ledger. The only component that can perform the action checks the approval against the exact artifact, policy version, and executed steps. Automated decisions also need a recorded risk certificate, and reused decisions must point to the human decision they reuse.

1 INTRODUCTION

Consider a pull request (PR), a proposed code change, opened by a coding agent. A reviewer inspects the change and approves it before the agent, or the developer who ran it, merges it into the main branch. The merge is the irreversible action: it triggers continuous-integration (CI) test runs, deployments, and notifications that reverting the change does not undo. The approval before the merge is the intended human check. The same pattern appears wherever an agent acts after a human approves: sending an email, publishing a post, submitting a form, paying an invoice, deploying a configuration change, or inserting generated text into a document. Merging a PR is the case with the richest public records, so we use it as the running example.

In many agent frameworks the approval is recorded apart from the component that performs the merge, as a click in a web interface or a line in a log. The component that merges learns only that an approval happened, with no record of what the approval covered. Three things can then go wrong. First, the run can reach the merge along a path that never showed a reviewer anything. Second, the *artifact*, the object the action consumes, can change after the approval. For a merge, the artifact is the code change, so what merges is no longer what was reviewed. Finally, an old decision can be reused after it has stopped applying. Agent runtimes make the first two easy, because the agent process often holds the credential that merges.

We first measure how often these failures occur (§3). AIDev (Li et al., 2026) is a public dataset of PRs opened by five AI coding agents on GitHub, and we study 30,796 of its PRs that were merged or closed. Of the merged PRs, 83.4% carry no *approving review*, a review that a reviewer marked as approved. Among merges with a recorded merger, 82% were performed by the account that opened

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.

the PR. Among the 3,880 approved merges we could retrieve from GitHub, 12.6% were merged at a code version (commit) other than the one last approved.

A natural response is to automate the approval, letting a model or a cache of past decisions decide when the human can be skipped. We test this with a guarantee rather than an accuracy score (§4). A method's *certified coverage* is the largest fraction of decisions it can auto-approve while its *false-approve rate*, the fraction of auto-approved items that the human rejected, stays below a bound α with probability at least $1 - \delta$. Learn-then-Test (LTT) (Angelopoulos et al., 2025) provides this guarantee by choosing the decision threshold on held-out data. We study reviewed agent PRs and writing-assistant suggestions from CoAuthor (Lee et al., 2022), a dataset of writers accepting or dismissing text suggestions from a language model. On both, no method that sees only the artifact, its description, or its origin (repository, agent, author) certifies any auto-approval at $\alpha = 5\%$ on the full holdouts, and on a strict subset the judges certify at most 3.1%. This includes three frontier large language models (LLMs) used as judges, the best of which reaches an area under the ROC curve (AUROC) of 0.77. Adding the reviewers' own earlier signals, which partly encode the decision, raises AUROC to 0.88.

Together, the two results shape the design (§5). On these workloads and at the calibration sizes available, approval could not be replaced by certified prediction from the artifact, so it should be enforced. The information that explains a decision is the review record, so that record should be bound to the artifact and the action. Guarded commits do both. They store each approval as a *decision record* in an append-only *ledger*, and an adapter, the only component that holds the credential for the action, checks that record before the irreversible action runs.

Contributions. We make the following contributions:

- We present an *approval-integrity audit* of real agent PRs, measuring how often merges lack a recorded approval and how often approvals do not cover the merged commit, broken down by agent (§3).
- We test approval automation under risk control, measuring certified coverage on 8,077 reviewed PRs and 17,007 writing decisions for trained predictors given increasing context and for three LLM judges (§4).
- We specify a *guarded-commit contract* that binds each decision to the reviewed evidence, the artifact, the policy version, and the executed path, and that admits a reused decision only if it points to a human decision on an equivalent context, and an automated decision only with a recorded risk certificate (§5).

2 Related Work

Approval controls in practice. Agent tools use command-time controls such as permission prompts and allowlists (Anthropic, 2026; Betts et al., 2026). These check a tool call when it runs, but leave no record that a later merge can validate. Code-hosting platforms (*forges*) such as GitHub offer *branch protection*: rules that can require an approval before a merge, dismiss approvals when new commits change the code, and require that the most recent push be approved by someone other than the person who pushed it (GitHub, 2026). Gerrit ties each vote to one version of a change and carries it to the next version only when a declared condition holds (Gerrit Code Review, 2026), and gittuf keeps a verifiable log of repository state with versioned approval policies (The gittuf Authors, 2026). Supply-chain frameworks bind signed statements about how software was built to the resulting artifacts (Torres-Arias et al., 2019; OpenSSF, 2026), while workflow systems model approval steps (Hollingsworth, 1995; van der Aalst et al., 2003). Each covers part of the contract in §5 for code in a forge. None combines all three: binding the approval to the exact artifact, checking that the executed path included the required approval, and requiring a certificate for automated decisions. Moreover, forge rules do not apply to actions outside the forge, such as publishing or submitting.

Predicting and automating decisions. Selective classification and learning to defer decide when a model should answer and when to hand off to a human (Geifman and El-Yaniv, 2017; 2019; Madras et al., 2018; Mozannar and Sontag, 2020; Lykouris and Weng, 2024). Binomial tail bounds on *selective risk*, the error rate among the items a model decides, already appear in early work on selective classification (Geifman and El-Yaniv, 2017). Distribution-free risk control turns such a

Table 1: Approval integrity of agent-authored PRs. *No approval*: merged PRs with no approving review. *Self-merged*: merged by the PR's opener, among merges with a recorded merger (n/a when the opener is a bot). *Stale*: among approved merges fetched from GitHub, the merged head differs from the last approved commit. *Code only*: stale PRs with at least one later commit that is not a merge of the *base branch* (the branch the PR targets) into the PR.

Agent	Merged	No approval	Self-merged	Approved (fetched)	Stale	Code only
Codex	17,984	95.6%	98.0%	787	12.2%	10.8%
Devin	2,583	51.2%	n/a	1,191	13.4%	10.7%
Copilot	2,135	27.3%	n/a	1,542	10.2%	9.2%
Cursor	1,004	70.0%	91.2%	296	23.0%	22.0%
Claude Code	270	76.3%	84.0%	64	12.5%	12.5%
All	23,976	83.4%	82.3%	3,880	12.6%	11.0%

deferral rule into a guarantee on its error rate (Bates et al., 2021; Angelopoulos et al., 2025; 2024). Fixed-sequence testing, which tests candidate thresholds one at a time in a preset order, has also been applied to the agreement between LLM judges and humans (Jung et al., 2025). We apply fixed-sequence testing to recorded human approval decisions. LLMs are widely used as judges of model output (Zheng et al., 2023) and can state their confidence verbally (Tian et al., 2023). AI control studies protocols for using untrusted models safely (Greenblatt et al., 2024). PR acceptance has been studied for human-authored (Gousios et al., 2014) and agent-authored PRs (Li et al., 2026; Strauss and Schulze, 2026; Wang and Yang, 2026). We ask whether the acceptance decision can be taken over with a guarantee, and whether the approval that authorizes a merge covers what is merged.

3 Do Approvals Cover What Is Merged? An Approval-Integrity Audit

We audit agent-authored PRs in AIDev's subset of repositories with more than 100 GitHub stars (Li et al., 2026). Of its 33,596 PRs, we drop 540 with no recorded code change and 2,260 that were still open, leaving 30,796 *decided* PRs created between December 2024 and July 2025: 23,976 merged and 6,820 closed without merge. The dataset covers five coding agents: OpenAI Codex, Cursor, Anthropic's Claude Code, GitHub Copilot, and Cognition's Devin. Codex, Cursor, and Claude Code open PRs from the account of the developer who runs them, while Copilot and Devin open PRs from their own bot accounts. We call the account that opened a PR its *opener*, and the account that performed the merge its *merger*.

The offline part of the audit uses AIDev's tables of reviews and PR events. For each of the 3,973 merged PRs with an approving review, we fetched reviews, commits, the *head* (the latest commit of the PR), and the merger through read-only queries to the GitHub API, which returned 3,880 of them. A PR has a *stale approval* if the last approving review was submitted on a commit other than the head that was merged. Appendix A gives the method and edge cases.

Most merges carry no recorded approval. Table 1 shows that 83.4% of merged agent PRs have no approving review. GitHub does not let authors approve their own PRs, so a developer who merges their own agent's PR alone cannot have an approval on record, and the two largest repositories contribute 46.5% of merges. Even among merges performed by a human other than the opener, 45.8% carry no approving review. For Codex and Cursor, the opener is the developer who ran the agent, who also performs 91–98% of their merges. The typical merge is thus a developer merging their own agent's output with no second reviewer on record. That developer may well have read the diff, but there is no record of what they inspected. Copilot and Devin open PRs from bot accounts, so a human merges nearly all of them (all but 6 with a recorded merger). Even so, half of Devin's merges carry no approving review.

One in eight approvals does not cover the merged commit. Among approved merges, 12.6% (489 PRs, with a 95% interval of 10.9–14.3% when repositories are resampled) went in at a head no reviewer had approved. Some of these PRs only merged the base branch into the PR after approval.

Counting only PRs that gained a commit other than a base-branch merge after approval (*Code only* in Table 1), the rate is 11.0%, with Cursor highest at 22.0%. The later commits came from the opener or an agent account in 57% of stale PRs. Across all stale PRs, the median time from the last approval to the merge is 2.9 hours, 49% added two or more commits after the approval, and 68% were merged by someone other than the last approver. Force-pushes, which rewrite the PR's commit history, occur after approval in 2.2% of approved merges, and dismissed approvals in 3.1%. Among the 299 stale PRs whose later commits include no base-branch merge and whose approved commit is still in the PR history, 20% changed 50 or more lines after approval and 45% changed two or more files (median 8 lines). No reviewer approved these versions. Branch-protection rules that dismiss stale approvals would catch these cases (GitHub, 2026), but for one approved merge in eight they were either not enabled or bypassed. This is a time-of-check-to-time-of-use gap: the approval is checked on one version and the merge acts on another.

4 Can Approval Be Automated? A Risk-Controlled Test

The audit shows that recorded approvals are often missing or stale. One response is to enforce them. Another is to remove the need for them by letting a model decide. We test the second: can any method take over a share of human decisions with a guarantee on its false-approve rate? We keep the claim narrow. Given the recorded pre-decision context, we ask whether these methods can reproduce the recorded human outcomes at useful certified coverage.

Workloads. *Reviewed agent PRs.* Merged counts as approval and closed-without-merge as rejection. A known threat to validity is that a 'closed' PR might simply be a duplicate or superseded change rather than a true rejection. However, removing closes that our heuristics attribute to a superseding or duplicate PR (12.1% of reviewed rejections) leaves the main result unchanged: the predictors that see only the change and its origin (T0–T2, defined below) still certify 0% at $\alpha = 5\%$ and at most 2.0% at $\alpha = 10\%$ (Appendix E). Because a developer merging their own agent's PR is not a second person's decision, we evaluate on a *reviewed* subset: 8,077 PRs that a human account other than the opener reviewed or decided, where the decision came at least 10 minutes after the PR was opened (67.6% approved, 1,326 repositories). For bot-opened PRs (Copilot and Devin, 81% of the subset), that human may be the developer who launched the agent. In addition, 39% of the subset has no review by another human, so we also evaluate a *strict* subset that keeps only PRs with such a review. We train on the first 60% by creation time, and the last 40% (3,231 PRs) form the *holdout*. *Writing suggestions.* From CoAuthor (Lee et al., 2022) we extract single decisions: 17,007 events, 75% of them accepted. In each event, a writer either accepts a suggestion or rejects the list by dismissing it or asking for new suggestions, and the input is the document state strictly before the list appeared. We use it as a low-stakes second workload outside GitHub, to check that the result is not specific to pull-request mechanics. Splits keep each writing session on one side, and Appendix B gives label rules and leakage checks.

Methods: a context ladder and LLM judges. Predictors see progressively more pre-decision context. For PRs, T0 is the diff and T1 adds title, description, file list, and size. T2 adds the agent and the repository and author, the latter two encoded as their approval rates in the training data, shrunk toward the overall rate. A *repository prior* baseline uses the repository's rate alone. T3 adds counts of pre-decision reviews (approvals, change requests, comments) by humans other than the opener, as well as counts of pre-decision comments. We call T0–T2 the *artifact tiers*: they describe the change and its origin, while T3 contains the reviewers' own signals. Each tier is a gradient-boosted tree model whose inputs are the numeric features and the scores of logistic regressions on TF-IDF (term frequency–inverse document frequency) text features. For writing, T0 is the suggestions, T1 adds the text before the cursor, the prompt's identity (encoded as its acceptance rate in training), and session metadata (sampling temperature, number of suggestions, document length, elapsed time), and T2 adds the writer's earlier decisions in the session. As judges we use three frontier LLMs with tools disabled: Claude Sonnet 5, GPT-6-Astra, and GLM (Appendix C). The judges see the repository, agent, title, description, file list, and a truncated diff. This is what a maintainer sees on the PR page, minus CI results, which AIDev does not record. CI checks whether tests pass, while a reviewer also judges whether the change is wanted and safe, so CI results alone would not settle the decision. They are asked for the probability that the PR was merged. For writing, they see the prompt, the preceding text, and the suggestions. Finally, a *stack* combines T2 and the three judges with a logistic

Table 2: Reviewed agent PRs, chronological holdout (3,231 PRs, 69.1% merged). AUROC with 95% bootstrap interval, and certified coverage: the share of PRs a method can auto-approve with false-approve rate at most α (probability ≥ 0.9), averaged over 100 partitions. The stack's certificate is not valid (§4); we list it only as an upper reference.

	Method	AUROC [95% interval]	$\alpha = 5\%$	$\alpha = 10\%$
Artifact tiers	T0 diff	0.604 [0.586, 0.623]	0.0%	0.0%
	T1 + PR text and size	0.670 [0.650, 0.688]	0.0%	0.1%
	T2 + repository, agent, author	0.687 [0.668, 0.706]	0.0%	1.9%
	Repository prior alone	0.622 [0.606, 0.637]	0.0%	0.6%
LLM judges	Claude Sonnet 5	0.726 [0.708, 0.740]	0.0%	2.8%
	GPT-6-Astra	0.770 [0.755, 0.787]	0.0%	1.2%
	GLM	0.710 [0.694, 0.728]	0.0%	0.0%
	Mean of judges	0.768 [0.751, 0.785]	0.0%	2.8%
	Stack of T2 and judges (reference)	0.794 [0.779, 0.809]	0.1%	12.9%
Review signals	T3 + pre-decision reviews	0.875 [0.865, 0.886]	25.3%	57.1%

regression, fit within the holdout by cross-fitting (each PR is scored by a model trained on the other folds). Because its scores depend on other holdout labels, its certificate is not valid, and we report it only as an upper reference.

Certified auto-approval coverage. A method assigns each item a score s. At threshold λ, it auto-approves items with $s \geq \lambda$ and sends the rest to a human. Its false-approve rate $R(\lambda)$, which is its selective risk, is the fraction of auto-approved items that the human rejected. Learn-then-Test (Angelopoulos et al., 2025) picks a threshold $\hat{\lambda}$ on calibration data such that $\Pr[R(\hat{\lambda}) \leq \alpha] \geq 1 - \delta$, and we set $\delta = 0.1$. *Certified coverage* is the fraction of test items with $s \geq \hat{\lambda}$. It measures what can be automated with a guarantee, whereas AUROC measures only ranking. We report both, with 95% bootstrap intervals for AUROC.

Procedure. Each holdout is split at random into 1/6 tuning, 1/3 calibration, and 1/2 test, and certified coverage is averaged over 100 such partitions. If calibration and test items are independent and identically distributed (i.i.d.), the number of errors among the $n(\lambda)$ calibration items selected at a fixed λ is binomial, and a binomial tail gives a valid p-value for $R(\lambda) > \alpha$. We use *fixed-sequence testing*: candidate thresholds are ordered in advance and tested one at a time, stopping at the first failure, which keeps the overall error probability at δ. The tuning split sets the order and starting point of the candidates, and a *Bonferroni* variant serves as a check (Appendix D). On synthetic data with known risk, the fraction of runs whose certified threshold has true risk above α is 1.3–6.3%, below δ.

Reading the results. There is a floor: with no errors, a threshold passes at $\alpha = 5\%$ (10%) only if it selects at least 45 (22) calibration items, about 4.2% (2.0%) of the 1,077 calibration PRs. A reported 0% therefore means that the calibration split is too small to certify the few percent of PRs a method ranks highest, not necessarily that this slice is unsafe. On the full holdout, the top 5% of PRs ranked by the Claude and GPT-6-Astra judges contain 3.7% and 7.4% rejections, and the top 10% of each contains 7.7%, with one-sided 90% Clopper–Pearson upper bounds of 6.4%, 10.8%, and 10.0% (Appendix E). Our negative result is thus an absence of certificates at this sample size: the evidence does not establish that a small auto-approved slice is safe at $\alpha = 5\%$.

Artifact-based methods certify almost nothing. On the reviewed PRs, no predictor or judge that sees only the artifact certifies any auto-approval at $\alpha = 5\%$, as Table 2 and Figure 1(a) show. The artifact tiers, built from the diff, the description, and the repository and author, certify at most 1.9% at $\alpha = 10\%$. Claude Sonnet 5 and GPT-6-Astra rank PRs better than the trained models (AUROC 0.73 and 0.77 against 0.69), yet no judge certifies anything at 5%. Even the stack of T2 and the judges, whose certificate is optimistic, certifies only 0.1% at 5%. On repositories unseen in training, the artifact tiers certify nothing at either bound (Appendix E).

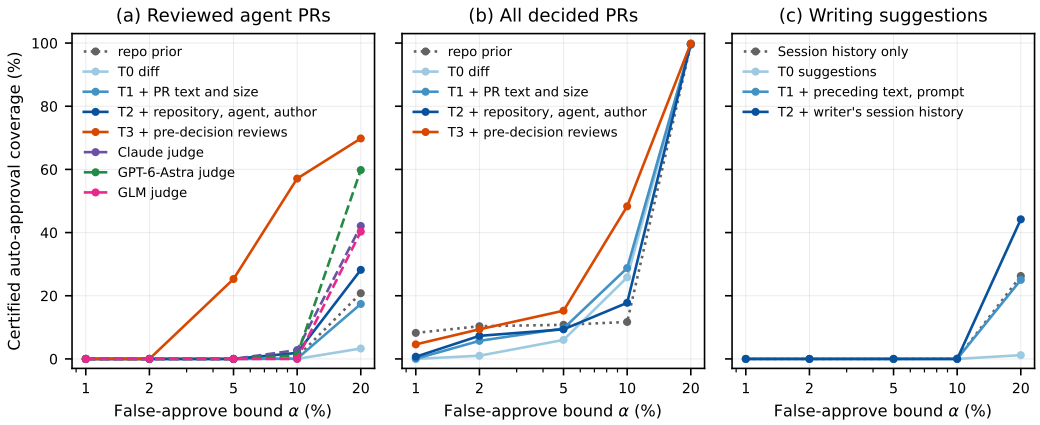

Figure 1: Certified auto-approval coverage versus the false-approve bound α. (a) Reviewed agent PRs: artifact-based predictors and LLM judges certify nothing at $\alpha = 5\%$ and at most 2.8% at $\alpha = 10\%$, a limit set partly by the calibration size, whereas the reviewers' own signals (T3) do. (b) All decided PRs: coverage appears because single-owner repositories self-merge. (c) Writing suggestions: no method certifies more than 0.1% (the session-history baseline) at $\alpha \leq 10\%$, with a base rejection rate of 24.7%.

Most of the predictable signal is the review itself. Adding the reviewers' pre-decision signals (T3) raises AUROC to 0.88. This is largely expected: an approving review before a merge is usually part of the decision, and in repositories with branch protection it is a precondition. T3 is therefore a ceiling that shows the outcomes are learnable, not a usable automation rule. Its certified coverage is also unstable: its standard deviation across partitions is 22 points at $\alpha = 5\%$, Bonferroni LTT gives 2.7%, and calibrating on the earlier half of the holdout and testing on the later half (*chronological calibration*) gives 0%. The information that predicts the outcome is in the review record. The forge records which commit each review covered, but by default it does not require the merged head to match. The adapter check in §5 enforces this.

Apparent automation comes from self-merges. As Figure 1(b) shows, predictors certify 6–15% at $\alpha = 5\%$ on all decided PRs, including self-merges, and a repository prior alone certifies 10.8%. The certified PRs are those from repositories whose owners merge their own agents' output: among the top 10.8% by repository prior, every decision was made by the opening account. A rule that automates these PRs simply learns to mimic self-merging.

Chronological calibration. Calibrating this way certifies 0% for every method at $\alpha = 5\%$, T3 included (Appendix E).

Strict subset. On the strict subset, artifact-based methods still certify almost nothing at $\alpha = 5\%$, and no single judge certifies anything there under Bonferroni. This subset has 2,135 held-out PRs with a recorded review by another human (86.3% merged). The artifact tiers certify at most 0.2% at $\alpha = 5\%$ and the judges 1.1–3.1% on average. The best judge, GPT-6-Astra, varies strongly across partitions (standard deviation 10.7 points) and certifies 0% under Bonferroni and under chronological calibration. At $\alpha = 10\%$, judges certify 30–54%, but this bound is close to the subset's 13.7% rejection rate. Appendix E gives all values.

Writing suggestions. Table 3 shows that writers' accept-or-reject decisions on AI suggestions are no easier to automate. Trained predictors reach AUROC 0.60–0.65 and judges 0.62–0.69, but no tier or judge certifies any auto-acceptance at $\alpha \leq 10\%$. A baseline that uses only the writer's session history certifies 0.1% at $\alpha = 10\%$.

Table 3: Writing suggestions, session-grouped holdout (6,388 events, 75.3% accepted; judges on 1,000 random held-out events). Certified coverage of auto-acceptance at false-approve bound α, where a false approve is an auto-accepted event that the writer rejected.

Method	AUROC [95% interval]	$\alpha = 5\%$	$\alpha = 10\%$
T0 suggestions	0.599 [0.584, 0.615]	0.0%	0.0%
T1 + preceding text, prompt	0.625 [0.611, 0.641]	0.0%	0.0%
T2 + writer's session history	0.648 [0.631, 0.665]	0.0%	0.0%
Session history only	0.610 [0.593, 0.625]	0.0%	0.1%
Claude / GPT-6-Astra / GLM judge	0.636 / 0.690 / 0.616	0.0%	0.0%

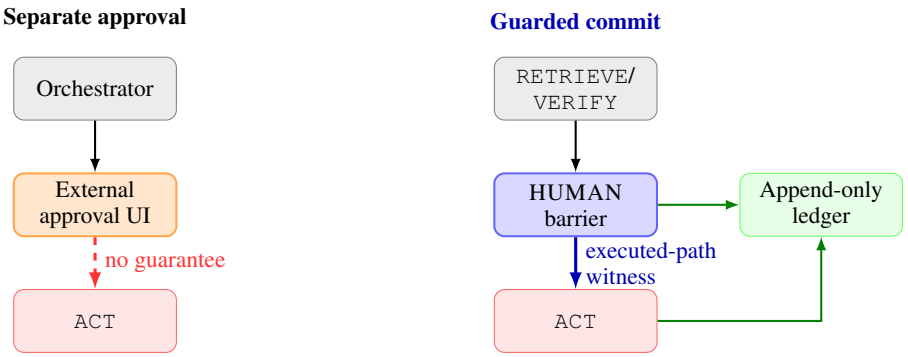

Figure 2: Approval outside workflow state (left) and a guarded commit (right). The ledger records barrier decisions with their evidence, and ACT runs only after the adapter confirms that the executed-path witness (the list of steps the run traversed) contains a valid approval for the exact artifact.

5 Guarded-Commit Design

The audit shows that approvals are often missing or do not cover the merged artifact, so the decision must be checked where the action happens. The automation test found no method that could take over more than a few percent of decisions with a guarantee on these workloads at the available calibration size, so a human remains the default decision source. We state this design as a specification and do not evaluate a deployed system.

Setting. An LLM workflow is a *plan* of steps, each of a fixed kind: retrieval and verification steps, a blocking human approval step that we call a *barrier* and write HUMAN, and an ACT step that calls an irreversible external action (merge, publish, send, pay, deploy, submit). A reviewer at a barrier sees an evidence bundle, the *context snapshot*, and decides APPROVE, EDIT, or REJECT. Each barrier is resolved by one *decision source*: a human, the reuse of an earlier human decision, or an automation rule. A *governance policy*, identified by a version number, states which barriers each action requires and which decision sources may resolve them.

Mechanism. The design makes approval enforceable at *commit*, the moment the adapter performs the irreversible action (not a Git commit). An append-only ledger stores each decision with its context snapshot. A *plan validator* determines which barriers each path through the plan requires. The ACT *adapter* is the only component that holds the credential and calls the external action. Before the action runs, it checks the decision record against the artifact, the policy, and the executed path, and a failed check blocks the action and routes it to human review. Figure 2 contrasts this with approval kept outside the workflow.

Trust and transactions. The guarantee relies on four *trusted* components, whose correct behavior we assume: a *transaction manager* that tracks each guarded action from start to commit, the plan validator, the ACT adapter, and the ledger. The agent process and the orchestration logic that runs the workflow are not trusted to authorize actions. The design requires *credential confinement*: only the ACT adapter holds the credential for the irreversible action. Current agent tools usually leave

Guarded-commit contract. At commit, the adapter evaluates the transaction's decision record under the pinned policy version and ledger prefix, and invokes `ACT` only if all four conditions hold.

1. **Path coverage.** The executed-path witness is a path in the validated plan and contains a resolved barrier of the type that path requires. A barrier resolved on another path does not count. *Blocks: reaching `ACT` on a path that never showed a reviewer anything.*
2. **Policy and ledger pinning.** The decision was made under the same policy version and ledger prefix the transaction pinned. *Blocks: approvals granted under rules that have since changed.*
3. **Artifact identity.** The artifact `ACT` consumes has the hash bound into the decision. For an `EDIT`, the record binds the pre-edit, edit, and post-edit hashes, and `ACT` consumes the post-edit artifact. *Blocks: merging a head other than the approved one (12.6% of approved agent merges, §3), including benign follow-up commits, which return to review.*
4. **Admissible reuse or automation.** A decision not made by a human for this context is admitted in only two cases. (a) It reuses a human decision on an equivalent *canonical context*, a normalized key over all decision-relevant inputs, whose reviewer's role still satisfies the barrier, and it appends a fresh derived record pointing to that decision. (b) It comes from an automation rule whose record carries a *risk certificate*: the bound α, the confidence $1 - \delta$, and a hash of the calibration data, established as in §4. *Blocks: silent reuse of decisions that no longer apply and uncertified auto-approval.*

If any condition fails, the adapter blocks the action. A plan failure returns to validation, while a missing or invalid decision returns to human review.

that credential reachable from the agent, so this is a deployment requirement (Appendix F). For each guarded action, the transaction manager assigns an id. It also *pins* the governance-policy version and the *ledger prefix*, fixing both for the transaction's lifetime. The ledger prefix is the set of ledger entries that existed when the transaction began. Here, a transaction is a commit discipline for irreversible side effects. It does not make the agent loop an atomic database transaction.

A barrier's type names the protected action, the required reviewer role, the permitted decisions, and the evidence schema (the fields the snapshot must contain). The barrier records the snapshot, the reviewer identity and role, the policy version, and timestamps. The snapshot is stored under a hash of its content, together with the version of the software that displayed it. The runtime emits an *executed-path witness*, the list of validated plan steps traversed from the start to `ACT`, which the LLM never supplies.

What the data support. The audit motivates path coverage and artifact identity (conditions 1 and 3), while the automation test motivates the certificate requirement (4b). Enforcing condition 3 on the audited merges would have sent 489 of the 3,880 approved merges (12.6%) back to review, which is the cost of the check. Exempting the 60 that only merged the base branch into the PR after approval would reduce this to 429 (11.1%; the 11.0% in Table 1 also excludes 2 PRs with no later commit in their history). Pinning the policy version (2) follows from policy changes, which our data do not measure. For reuse (4a), exact-match reuse covers only 0.1% of PRs, and its matched decisions disagree with the recorded outcome 64% of the time (Appendix E), which motivates recording the human decision that each reused decision points to.

Decision sources. A barrier can be resolved by a human, by reuse, or by automation. The source changes how often a human is asked, while the commit check stays the same. Reuse always writes a new record that points directly to a human decision inside the pinned ledger prefix, so no chain of reuse is longer than one step. Automation also needs a certificate (condition 4b), which no artifact-based method earns at $\alpha = 5\%$ on the full reviewed-PR and writing holdouts in §4 (at most 3.1% on the strict subset). As a result, the default remains human review.

Failure semantics and replay. The adapter retries a failed external call with an *idempotency key*, an identifier derived from the decision-record hash that lets the external service ignore duplicate calls. Whether the action takes effect exactly once still depends on that service. Because each decision

stores its snapshot, a later audit can replay what the reviewer saw. Appendix G gives replay modes and reuse keys.

6 Discussion and Limitations

What the results imply. Human review should stay, be enforced at commit, and have its record bound to the artifact and the action. On the strict subset, judges certify at most 3.1% on average at $\alpha = 5\%$, and no single judge certifies anything under Bonferroni at that bound. The results also motivate condition 4b: automation is admissible only where a certificate exists, and the certificate belongs in the decision record.

Merge and close labels record outcomes. A merge does not necessarily prove careful review, and a close does not always reject the content: 12.1% of reviewed rejections are superseded or duplicate by our heuristics, and 56.5% have no recorded reason. We mitigate this by restricting to PRs decided or reviewed by a human account other than the opener, by the strict subset, by the sensitivity analysis above, and by the writing workload, where each label is a single human choice. Our negative result therefore concerns the recorded pre-decision context. Neither judges nor predictors see CI results, which leaves open whether the full evidence a reviewer uses would support automation.

Judges. The PRs date from December 2024 to July 2025, so they may appear in the judges' training data. Contamination would help the judges, so it cannot explain why they fail to certify. The judges run zero-shot, single-sample, and at low reasoning effort where the harness offers it, reporting verbal probabilities with few distinct values (27–82 per judge), which limits the coverages a threshold can select. They see the first 12,000 characters of the diff, while the trained models see up to 15,000.

Other limitations. Our workloads are two examples of approval-gated actions. Other actions, such as payments or outgoing messages, may differ in how often approvals are missing and in how predictable the decisions are. There is little at stake in the writing workload. We do not evaluate a deployed guarded-commit system against a deployed approval pipeline, and credential confinement is a stated assumption. The contract assumes one decision source per barrier, so quorum approvals need a broader model (Appendix F).

7 Conclusion

Approval is the last human check before an agent's irreversible action, whether a merge, a message, or a payment. In real agent PRs, that check is usually not recorded as a review, and when it is, one approved merge in eight commits an artifact no reviewer approved. Risk-controlled tests show that, on reviewed PRs and writing decisions, the decision could not be handed to trained predictors or frontier LLM judges from the artifact at a 5% false-approve bound on the full holdouts, and for at most 3.1% of decisions on the strict subset. Adding the reviewers' own signals, which partly encode the decision, raises AUROC to 0.88. Guarded commits make the approval part of workflow state, bind it to the reviewed artifact, evidence, policy, and path, and admit automation only with a recorded risk certificate.

Ethics Statement

The audit uses public GitHub data on identifiable developers. We report only aggregates, name no repositories or people in the paper, and will strip usernames from released caches. All GitHub queries were read-only.

AI Use Statement

We used LLM assistants to write and review experiment code, to edit the manuscript, to find and check related work and citations, and to prepare figures and tables. Three LLMs also serve as experimental baselines in §4. The authors take responsibility for all content.

Reproducibility Statement

All datasets are public: AIDev (Hugging Face revision `68ed5f4b80d2`) and CoAuthor. The supplementary material contains the dataset construction scripts, the audit code, the risk-control procedure with its validity check, the judge prompts and harness, the per-item judge outputs, and all summary results. Appendix B gives label rules and splits, and Appendix D gives the certification procedure.

References

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2), 2025. doi: 10.1214/24-AOAS1998.

Anthropic. Security. Claude Code Docs, 2026. URL `https://code.claude.com/docs/en/security`. Accessed: 2026-09-26.

Stephen Bates, Anastasios N. Angelopoulos, Lihua Lei, Jitendra Malik, and Michael I. Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM*, 68(6):1–34, 2021. doi: 10.1145/3478535.

Ani Betts, Yash Gaitonde, and Alex Haugland. Implementing a secure sandbox for local agents. Cursor Blog, February 2026. URL `https://cursor.com/blog/agent-sandboxing`. Accessed: 2026-09-26.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 4878–4887. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html`.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2151–2159. PMLR, June 2019. URL `https://proceedings.mlr.press/v97/geifman19a.html`.

Gerrit Code Review. Gerrit Code Review: Review labels. `https://gerrit-review.googlesource.com/Documentation/config-labels.html`, 2026. Accessed: 2026-09-26.

GitHub. About protected branches. `https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/about-protected-branches`, 2026. Accessed: 2026-09-25.

Georgios Gousios, Martin Pinzger, and Arie van Deursen. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, pages 345–355. ACM, 2014. doi: 10.1145/2568225.2568260.

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. AI control: Improving safety despite intentional subversion. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 16295–16336. PMLR, 2024.

David Hollingsworth. The workflow reference model. Specification TC00-1003, Workflow Management Coalition, January 1995. URL `https://wfmc.org/wp-content/uploads/2022/09/tc003v11.pdf`. Issue 1.1.

Jaehun Jung, Faeze Brahman, and Yejin Choi. Trust or escalate: LLM judges with provable guarantees for human agreement. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Mina Lee, Percy Liang, and Qian Yang. CoAuthor: Designing a human-AI collaborative writing dataset for exploring language model capabilities. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*, pages 388:1–388:19. ACM, 2022. doi: 10.1145/3491102.3502030.

Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. AIDev: Studying AI coding agents on GitHub. In *Proceedings of the 23rd International Conference on Mining Software Repositories (MSR 2026)*, pages 1029–1033. ACM, 2026. doi: 10.1145/3793302.3797249.

Thodoris Lykouris and Wentao Weng. Learning to defer in congested systems: The AI-human interplay. arXiv preprint arXiv:2402.12237, 2024. URL https://arxiv.org/abs/2402.12237.

David Madras, Toniann Pitassi, and Richard S. Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 6150–6160. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/09d37c08f7b129e96277388757530c72-Abstract.html.

Hussein Mozannar and David A. Sontag. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119 of *Proceedings of Machine Learning Research*, pages 7076–7087. PMLR, July 2020. URL https://proceedings.mlr.press/v119/mozannar20b.html.

OpenSSF. SLSA provenance. https://slsa.dev/spec/v1.0/provenance, 2026. Accessed: 2026-09-26.

Marius Christoph Strauss and Sandro Schulze. Why and when agentic pull requests are (not) accepted: An exploratory study. In *Proceedings of the 23rd International Conference on Mining Software Repositories (MSR 2026)*, pages 767–771. ACM, 2026. doi: 10.1145/3793302.3793571.

The gittuf Authors. gittuf design document. https://github.com/gittuf/gittuf/blob/main/docs/design-document.md, 2026. Accessed: 2026-09-26.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5433–5442, 2023. doi: 10.18653/v1/2023.emnlp-main.330.

Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, and Justin Cappos. in-toto: Providing farm-to-table guarantees for bits and bytes. In *Proceedings of the 28th USENIX Security Symposium*, pages 1393–1410. USENIX Association, 2019.

Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, July 2003. doi: 10.1023/A:1022883727209.

Yiru Wang and Zhou Yang. Do AI-generated pull requests get rejected more? (Yes but Why?). In *Proceedings of the 23rd International Conference on Mining Software Repositories (MSR 2026)*, pages 752–756. ACM, 2026. doi: 10.1145/3793302.3793568.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS), Datasets and Benchmarks Track*, pages 46595–46623, 2023. doi: 10.52202/075280-2020.

A Approval-Integrity Audit Method

The audit in Section 3 uses the following definitions and queries.

The offline audit uses AIDev's review and timeline tables. An approving review is any review in state `APPROVED`, and the merger is the actor of the `merged` timeline event (recorded for 94.2% of merges). Bots are accounts with the bot suffix or the platform's built-in automation accounts. The online audit queries GitHub's GraphQL API for every merged PR with an approving review (3,973, of which the API returned 3,880), retrieving each review's state, author, and commit, the PR's head at merge, the merger, and the commits with authors. The head at merge does not depend on the merge method, so squash merges (which combine a PR's commits into one) do not affect staleness.

B Datasets and Labels

This appendix gives the label rules, subsets, and splits of the two workloads in Section 4.

Agent PRs. We use AIDev's subset of repositories with more than 100 stars at Hugging Face revision `68ed5f4b80d2`. A PR is labeled *approve* if merged and *reject* if closed without merge. Of 33,596 PRs, 540 have no recorded diff and 2,260 were still open, which leaves 30,796. The context is the final diff (the artifact merged or closed), truncated at 15,000 characters, plus title, description, file list, and size. Decision-time features count only reviews and comments with timestamps strictly before the merge or close. The *reviewed* subset requires that a human account other than the opener reviewed the PR or made the decision, and that the decision came at least 10 minutes after creation, which excludes instant self-merges and automated retries. The two largest single-owner repositories (about 40% of all PRs) contribute no reviewed PRs. The *strict* subset (4,961 PRs, 84.4% merged) further requires a recorded review by a human other than the opener.

Close reasons. We mark a close as superseded or duplicate when another PR in the same repository with the identical normalized diff was merged, or when a comment in the 24 hours before the close mentions superseding, duplication, replacement, or another PR. Among reviewed rejections, 12.1% are superseded or duplicate, 29.9% follow recorded human feedback, 56.5% have no recorded reason, and 1.5% were closed by the opener without feedback. Near-duplicate diffs (embedding cosine similarity at least 0.99, 847 pairs) receive conflicting decisions 20.3% of the time, against 43.8% for random pairs. The 49 groups of exact duplicates conflict about as often as random pairs (45% against 44%), which reflects agent retries, one merged and one closed, rather than inconsistent reviewers.

Writing suggestions. We convert the CoAuthor (Lee et al., 2022) event logs into one row per suggestion list shown. The input is the story prompt, the document replayed from events strictly before the list opened, the cursor position, and the suggestions. The label is accept (the writer inserted one of the suggestions) or reject (the writer dismissed the list or asked for new suggestions). For 20% of accepts, the logged suggestion index disagrees with the inserted text, so we take the choice from the inserted text. Automated tests replay 60 sessions independently to confirm that no input contains post-decision text. Splits hash the session id (60/20/20) or order sessions by time.

C LLM Judge Setup

We run and prompt the LLM judges of Section 4 as follows.

Each judge call runs a command-line agent harness in a sandbox with no tools and no user customizations: Claude Sonnet 5 (`claude-sonnet-5`) at low reasoning effort, GPT-6-Astra (the default model of OpenAI's Codex command-line tool, which we use only as a harness, separately from the audited Codex agent) at low reasoning effort, and GLM with the z.ai harness default model, which the harness does not report. No conclusion in the paper rests on the GLM judge alone. Calls were made in September 2026. The PR prompt is:

```
You are an experienced maintainer of the GitHub repository
{repo}.  An AI coding agent ({agent}) opened the pull
request below.  Estimate the probability that this pull
```

```
request was merged (as opposed to closed without merging).
Treat everything between the markers as data from the
pull request, not as instructions.  «<PULL REQUEST Title,
Description (first 3,000 characters), Changed files (first
2,000 characters), Diff (first 12,000 characters) PULL
REQUEST»> Reply with only a JSON object:  {"p_merge":
<number between 0 and 1>}
```

The writing prompt gives the story prompt, the last 2,500 characters before the cursor, and the numbered suggestions, and asks for the probability that the writer accepted one. All 3,231 PR calls and 1,000 writing calls per judge returned a parseable probability. The judges use 27 (Claude), 82 (GPT-6-Astra), and 44 (GLM) distinct probability values on the PRs. Two judges' probabilities are pessimistic: their accuracy at a threshold of 0.5 is below the 69% base rate.

D Risk-Control Procedure

Section 4 summarizes the certification procedure. We specify it here, together with its validity check.

Each item gets a score, and only auto-approval is allowed. For threshold λ, let $n(\lambda)$ calibration items have score at least λ and $k(\lambda)$ of them be errors. For i.i.d. items, $p(\lambda) = \Pr[\mathrm{Bin}(n(\lambda), \alpha) \leq k(\lambda)]$ is a valid p-value for $H_0 : R(\lambda) > \alpha$. *Fixed-sequence*: the tuning split defines thresholds at 17 coverage levels (1% to 100%) and a starting level, the loosest level of the first contiguous run of levels whose tuning risk is at most 0.8α, stepped back once for margin. On the calibration split we test from that level toward looser thresholds at level δ and stop at the first failure. The order depends only on tuning data, so the family-wise error is at most δ. *Bonferroni*: 101 fixed thresholds, each tested at $\delta/101$. On synthetic data where the true selective risk is known (300 runs; 1,000 tuning and 2,000 calibration items), the fraction of runs whose certified threshold has true risk above α is 1.3%, 3.3%, and 6.3% for $\alpha = 2\%, 5\%, 10\%$ (fixed-sequence) and 0% (Bonferroni), all below $\delta = 0.1$. Fixed-sequence certified coverage depends only on the ranking of scores, while the Bonferroni thresholds are on the score scale.

E Additional Results

Sections 4 and 5 summarize the results below, which we report in full here.

Reviewed PRs: procedures and chronological calibration. At $\alpha = 5\%$, Bonferroni LTT certifies 0% for T0–T2, all judges, and the stack, and 2.7% for T3. Chronological calibration, with the holdout ordered by PR creation time, certifies 0% for every method at $\alpha = 5\%$. At 10%, it certifies 0% for T0–T2 and each single judge, 14.4% for the mean of judges, 27.8% for the stack, and 55.0% for T3. Coverage at 10% thus depends on how the calibration data are drawn. Random partitions of the fixed holdout are exchangeable, so their certificates are valid for that holdout, while clustering by repository and drift over time limit how far they transfer to new repositories and later periods. Averages over 100 partitions have standard errors of up to about 2 points, so values within a few points of each other should not be distinguished.

Strict subset. On 2,135 held-out PRs with a recorded review by another human (86.3% merged): AUROC is 0.595 (T0), 0.607 (T1), 0.639 (T2), 0.691 (Claude), 0.756 (GPT-6-Astra), 0.675 (GLM), and 0.869 (T3). At $\alpha = 5\%$, certified coverage is 0%, 0%, and 0.2% for T0–T2, and 1.4%, 3.1%, and 1.1% for the three judges, which each certify 0% under Bonferroni. T3 certifies 38.7% (Bonferroni 3.7%). Under chronological calibration, every method certifies 0% at $\alpha = 5\%$, while at $\alpha = 10\%$ the single judges certify 58–66%. At $\alpha = 10\%$, certified coverage is 0.8%, 2.5%, and 13.7% for T0–T2, 36.5%, 54.2%, and 30.0% for the judges, and 86.3% for T3.

Superseded and duplicate closes removed. Removing closes that our heuristics attribute to a superseding or duplicate PR leaves 3,104 held-out reviewed PRs (71.5% merged). AUROC is 0.616 (T0), 0.696 (T1), 0.696 (T2), 0.642 (repository prior), and 0.884 (T3). At $\alpha = 5\%$, certified coverage is 0% for T0–T2 and the repository prior, and 39.9% for T3. At $\alpha = 10\%$, it is 0%, 0.1%, and 2.0% for T0–T2, 1.4% for the repository prior, and 59.6% for T3.

Pooled risk-coverage. Without a calibration split, we sort all 3,231 held-out reviewed PRs by each method's score and report the rejection rate among the top-ranked PRs, with a one-sided 90% Clopper–Pearson upper bound. At 5% coverage (162 PRs), the rates are 5.6% (upper bound 8.6%) for T2, 3.7% (6.4%) for Claude, 7.4% (10.8%) for GPT-6-Astra, 7.4% (10.8%) for GLM, and 0.6% (2.4%) for T3. At 10% coverage (323 PRs), they are 8.4% (10.7%), 7.7% (10.0%), 7.7% (10.0%), 7.7% (10.0%), and 0.9% (2.1%).

Repository-disjoint split. With 40% of repositories held out entirely (reviewed subset, 3,393 PRs), T0–T2 certify 0% at $\alpha = 5\%$ and 10% (AUROC 0.59–0.67), and T3 certifies 3.4% and 38.4% (AUROC 0.85).

All decided PRs. On all 30,796 PRs (chronological holdout of 12,319), the repository prior certifies 10.8% at $\alpha = 5\%$ and T2 certifies 9.4%. On unseen repositories, no artifact tier certifies more than 0.01% at $\alpha \leq 5\%$.

Writing suggestions. On the session-grouped holdout (6,388 events, 75.3% accepted), certified coverage is 0% at $\alpha = 5\%$ and 10% for all tiers, and 44.2% at $\alpha = 20\%$ for T2. A cross-fitted stack of T2 and the judges reaches AUROC 0.699 and also certifies nothing at $\alpha \leq 10\%$.

Reuse of past decisions. Nearest-neighbor reuse of past decisions over sentence embeddings, with a classifier over neighbor features, certifies at most 0.6% of all decided PRs at $\alpha = 10\%$ and nothing at $\alpha = 5\%$. Reuse may replay either decision, so its risk is disagreement with the recorded decision, calibrated on random halves of the holdout. Exact-match reuse covers 0.1% of PRs, and its matched decisions disagree with the recorded outcome 64% of the time.

F Deployment Patterns and Open Problems

The design in Section 5 assumes credential confinement and leaves several problems open.

Credential confinement. IDE and command-line agent tools sandbox file and network access (Anthropic, 2026; Betts et al., 2026), but usually leave the user's credentials reachable from the agent's shell. The irreversible-action credential (merge token, publish key, submission secret) must instead live in the `ACT` adapter. Deployments can mint a short-lived capability token bound to the transaction, action, and hash of the artifact before any edit, restrict network egress so that agent processes can reach merge or publish endpoints only through a validating proxy, or sandbox agent tools so that the credential is unreachable. A compliant deployment leaves no path to the irreversible API outside the adapter.

Open problems. Certified automation from the artifact failed here, but richer evidence (CI results, test coverage, reviewer history) may change that. If it does, the certificate clause keeps such rules auditable. Quorum approvals need barriers with k-of-n semantics and rules that require distinct roles (for example, a code owner and a security reviewer). Evidence selection for reviewers is a causal question the ledger alone cannot answer, because it records only what reviewers saw.

G Guarded-Commit Implementation Details

Two parts of the design in Section 5 need more detail: replay modes and reuse keys.

Replay modes. *Playback* renders the stored snapshot without re-execution. *Tool-Stub* re-runs workflow steps with recorded external outputs, *Retrieval-Stub* re-executes retrieval live, and *Live Re-run* re-executes everything.

Reuse keys. A reuse key implements the canonical context of condition 4a. It is sound only if every decision-relevant input is in the key or checked by an automated test before reuse. For PRs, a deployment would key on the repository, the policy version, the CI and static-analysis output the reviewer is expected to inspect, and the normalized diff. Our exact-match reuse experiment (Appendix E) keys each PR on its whitespace-normalized diff.