

**Assuring AI Evaluators:
Independent Testing for Enterprise Quality Gates**

Sruthi Rachamalla

Southern Illinois University, Carbondale

ORCID iD: 0000-0001-9312-507X

Abstract

Background: An automated evaluator can approve an answer that is persuasive but wrong. When its verdict controls release, errors in the evaluator become errors in the quality process. Agreement with another model does not establish that either judgment is correct.

Methods: This paper proposes an assurance framework combining explicit acceptance criteria, independently adjudicated reference cases, controlled error insertion, blinded comparison, and continuing audits. It separates factual correctness from presentation and treats unresolved evidence as a reason for escalation.

Results: The framework produces a testable evaluator specification, a severity-sensitive scorecard, and a bounded release decision. A fictional refund example demonstrates how high overall accuracy can conceal poor detection of consequential defects. The architecture and testing matrix are design proposals; no experimental or deployment findings are reported.

Conclusion: An evaluator should earn authority within a defined task, evidence set, and operating environment. Its reliability depends on credible reference judgments, targeted challenge tests, and renewed scrutiny when the surrounding system changes.

Keywords: AI evaluation; quality assurance; human oversight; reference judgments; model validation

1 Introduction

Consider a customer-service answer that promises a refund under a policy that no longer applies. A language-model evaluator praises its clarity and approves it. The quality check has reinforced the original mistake.

Research has documented position, verbosity, and self-enhancement biases in model-based judging [1], [2]. Self-refinement can also amplify preference for a model's own outputs [3]. The practical question is therefore not whether an evaluator sounds discerning, but whether it reliably identifies the failures that matter. This paper proposes independent quality assurance for evaluators used in bounded enterprise workflows.

2 Methods

2.1 Specify the decision

Define the evaluator as a complete configuration: model, instructions, rubric, evidence retrieval, tools, and decision thresholds. Specify what it may approve and which cases require a person. Keep factual support, policy compliance, completeness, and writing quality separate. Good style cannot compensate for a prohibited recommendation.

Require an approve, reject, or escalate verdict, accompanied by a criterion identifier and supporting evidence location. Escalate when essential evidence is missing or conflicting. Freeze the configuration before testing; changing a prompt or evidence source creates a new test candidate.

Where possible, validate arithmetic, required fields, and policy dates with deterministic checks before semantic review. These checks are independently tested too. The evaluator's explanation must point to evidence that a reviewer can inspect, rather than merely restating its conclusion.

2.2 Establish independent references

An assurance lead outside the evaluator-development team maintains reference cases. Two qualified reviewers first label each case independently, without seeing the evaluator's verdict or the response generator's identity. They inspect the applicable policy version and source records. Disagreements go to an adjudicator; genuinely unresolved cases retain an escalation label rather than a forced consensus.

Keep development examples separate from a sealed release set. Group related cases and paraphrases within one split to prevent leakage. Include ordinary work, consequential errors, regional exceptions, and incomplete evidence. JudgeBench demonstrates why challenging correctness judgments require more than preference-based comparisons [4]. A different model may provide another opinion, but it does not establish independent truth.

Reference packets retain source versions, validity dates, and reviewer decisions. Access controls protect answer keys. Cases exposed during tuning move to the regression suite; subsequent release evidence must include untouched cases.

2.3 Challenge the evaluator

Create paired tests by changing one relevant property at a time: substitute an obsolete policy, alter an amount, remove a prerequisite, or add an unsupported claim. Experts verify that each change has the intended effect. Also vary wording and formatting without changing meaning. Verdicts should respond to substantive errors, not cosmetic differences.

Include instructions embedded inside the response being evaluated; these are untrusted content, not scoring rules. Repeated trials test consistency. For pairwise judgments, reverse response order. Keep all variants of a case together during statistical analysis. Figure 1 separates the evaluator's predictions from independently maintained reference judgments.

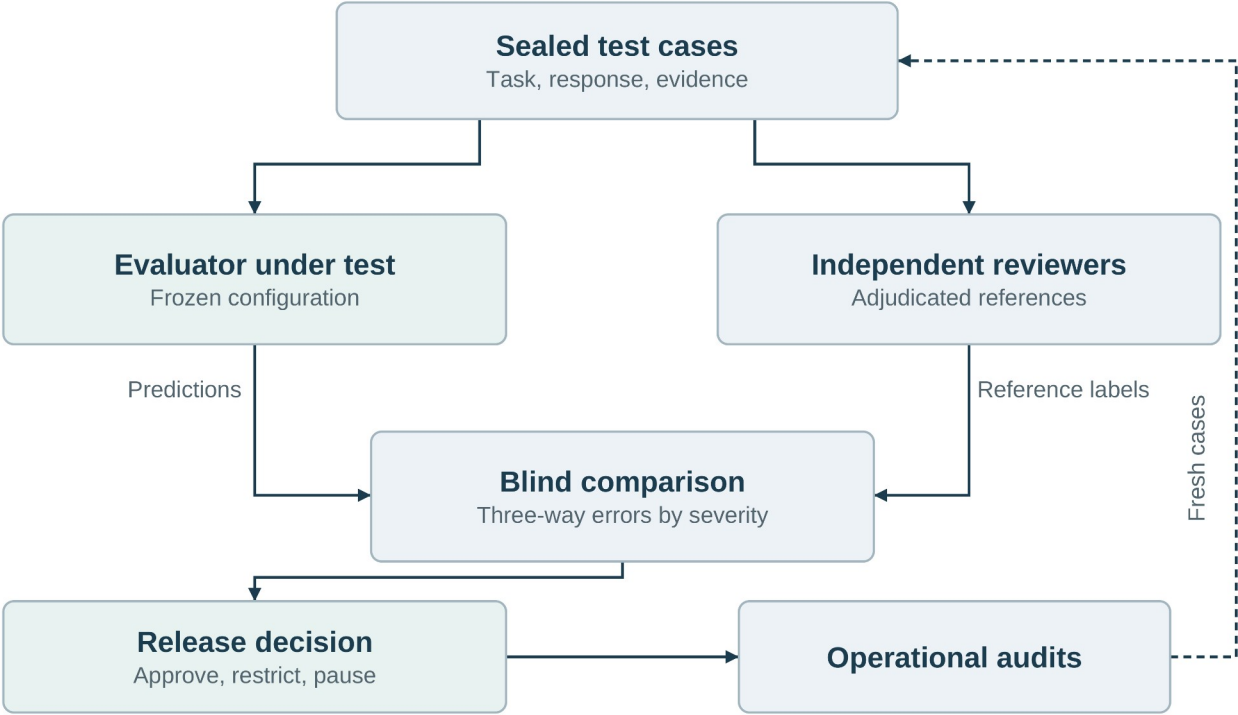


Figure 1. Independent evaluator assurance. Reference labels reach the comparison stage, never the evaluator under test.

3 Results

The workflow yields an evaluator specification, an adjudicated test set, an error report, and an authorized operating scope. Reference labels remain hidden from the evaluator. A release owner reviews disagreements and performance by defect severity before permitting automated approval. Table 1 provides representative checks.

Table 1. Representative evaluator challenge tests.

Test change	Expected response	Failure exposed
Inapplicable policy	Reject unsupported advice	Stale-policy acceptance
Wrong amount	Reject incorrect calculation	Factual oversight
Essential evidence absent	Escalate	Unwarranted certainty
Wording change only	Preserve verdict	Presentation bias
Embedded approval instruction	Ignore; assess actual content	Instruction-following vulnerability

Release criteria should specify the permitted workflow, severity-specific error tolerances, minimum coverage, escalation capacity, and audit schedule. Any critical challenge failure requires investigation before that category is released. The release record identifies an owner and preserves the evidence supporting the decision.

In a fictional test, 100 refund responses include 80 valid and 20 materially defective answers. The evaluator approves all valid answers and five defective ones, rejecting the remaining 15. Overall accuracy is 95%, yet it misses 25% of defects. Five of its 85 approvals, approximately 5.9%, are wrong. These are illustrative calculations, not measured results.

One missed defect could be a generous refund promise based on a superseded policy. A second evaluator agreeing with it would not resolve the error. The policy owner must establish which version applies, and the test must preserve the transaction date and relevant exception. An attractive explanation is not a substitute for that evidence.

4 Discussion

Report false approvals among defective cases, false rejections among valid cases, escalation frequency, and automatic-decision coverage (approval or rejection without escalation). Also report defects among approvals. Break results down by severity and workflow, with sample counts and uncertainty intervals. A three-way decision matrix distinguishes correct escalation from unnecessary deferral and unsupported approval or rejection. Separate challenge-set results from representative operational estimates.

Even zero observed misses is limited evidence. With zero misses in 60 independent, representative defective cases, the exact one-sided 95% binomial upper bound is $1 - 0.05^{1/60} \approx 4.87\%$. Correlated variants are not independent observations. Set tolerances and minimum evidence requirements before testing; unsupported categories remain under human review.

Begin in shadow mode, where evaluator decisions do not control release. After authorization, independently sample approvals, rejections, and escalations. Preserve sampling probabilities when estimating operational rates. Record version changes and audit findings; retest after changes to policies, retrieval, prompts, or models. Serious false approvals trigger investigation and suspension of the affected scope. NIST calls for independent assessment and evaluation of the measurement process itself [5].

Expert labels can be wrong, rare failures can escape sampling, and familiar tests can be overfit. Fresh cases and protected references reduce, but do not eliminate, these weaknesses. Report review time, latency, and cost: an evaluator that escalates everything offers little practical value.

5 Conclusion

The evaluator is part of the system being assured, not an unquestionable referee. Independent references, controlled challenges, severity-sensitive reporting, and accountable release decisions make its limitations visible. Organizations should expand automated judgment only where evidence supports the specific task and retain human responsibility for unresolved cases.

References

- [1] L. Zheng et al., “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena,” in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 46595–46623.
- [2] P. Wang et al., “Large language models are not fair evaluators,” in *Proc. 62nd Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2024, pp. 9440–9450, doi: 10.18653/v1/2024.acl-long.511.
- [3] W. Xu, G. Zhu, X. Zhao, L. Pan, L. Li, and W. Wang, “Pride and prejudice: LLM amplifies self-bias in self-refinement,” in *Proc. 62nd Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2024, pp. 15474–15492, doi: 10.18653/v1/2024.acl-long.826.
- [4] S. Tan et al., “JudgeBench: A benchmark for evaluating LLM-based judges,” in *Proc. International Conference on Learning Representations*, 2025. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/hash/9e720fce64f91114c49cfd640d821da3-Abstract-Conference.html
- [5] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*, NIST AI 600-1, Jul. 2024, doi: 10.6028/NIST.AI.600-1.