

Auditable Acceptance of LLM-Generated Code: An Evidence-Based Testing Architecture and Multi-Source Evaluation

Zhiyuan Li

Southwest Jiaotong University

Li Feng

fenglifeng@swjtu.edu.cn

Southwest Jiaotong University

Yurui Feng

University of Bristol

Chaofan Ji

Southwest Jiaotong University

Jian Xu

Southwest Jiaotong University

Chen Chen

Sichuan Agricultural University

Research Article

Keywords: Large language models, software testing, intelligent testing infrastructure, verification evidence, evidence-conditioned triage, acceptance-risk evaluation

Posted Date: October 21st, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11177090/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Auditable Acceptance of LLM-Generated Code: An Evidence-Based Testing Architecture and Multi-Source Evaluation

Zhiyuan Li¹, Li Feng^{1*}, Yurui Feng², Chaofan Ji¹, Jian Xu¹,
Chen Chen³

¹School of Computing and Artificial Intelligence, Southwest Jiaotong University, No. 999 Xi'an Road, Pidu District, Chengdu 611756, China.

²School of Computer Science, University of Bristol, Merchant Venturers Building, 75 Woodland Road, Bristol BS8 1UB, United Kingdom.

³Business and Tourism School, Sichuan Agricultural University, Dujiangyan, China.

*Corresponding author(s). E-mail(s): fengli@swjtu.edu.cn;

Contributing authors: swjtu2025@my.swjtu.edu.cn;

qs22234@bristol.ac.uk; jichaofan@my.swjtu.edu.cn;

1115304250@qq.com; chenchen6@stu.sicau.edu.cn;

Abstract

Accepting generated code requires a record of verification evidence and a distinction between detected violations and unresolved uncertainty. We present Intelligent Testing Infrastructure (ITI), an architecture linking verification results to candidate artifacts and recording Accept–Escalate–Reject decisions before offline adjudication. We examine 80 constructed Python candidates, public EvalPlus scores, TC-Bench detection matrices, and 200 Qwen3:8b and DeepSeek candidates on 50 LiveCodeBench tasks. Controlled adequacy-based triage reduces false acceptance under a composite correctness criterion from 21/59 to 11/59. Correct-candidate retention falls proportionally, leaving accepted-set error risk at 50%. In the two-model study, ITI reproduces always-on strong verification decisions with 132 rather than 200 strong-verifier invocations: 34% fewer invocations and 35.9% less attributed verification time within the shared execution trace. ITI shows no improvement in proxy-labelled accepted-set composition over size-matched random retention; all compared strategies accept the same DeepSeek candidates. Neither candidate-level study provides independent correctness adjudication: the controlled criterion incorporates the evaluated tests,

and the development study uses Qwen review labels. The findings demonstrate selective escalation in the evaluated workflow and illustrate how recorded evidence supports review of acceptance decisions. Whether ITI improves correctness discrimination requires a confirmatory study with independent adjudication and tasks held out from policy development.

Keywords: Large language models , software testing , intelligent testing infrastructure , verification evidence , evidence-conditioned triage , acceptance-risk evaluation

1 Introduction

Large language models now support code completion, function synthesis, repository modification, and multi-step software-engineering tasks (Fan et al. 2023; Jimenez et al. 2024; Yang et al. 2024). Generated artifacts may be selected or integrated with limited human inspection, making the interpretation of test results important. A successful test run shows that a particular verifier found no violation within its scope, environment, and oracle. It does not establish correctness across all relevant behaviors or, by itself, justify automatic acceptance.

Finite tests cover only part of a program’s possible behavior. Their effectiveness depends on input selection, execution conditions, and oracles (Zhu et al. 1997; Barr et al. 2015). Additional verification can expose errors missed by a basic suite, so an observed pass reflects both the candidate and the checks applied to it.

Pass/fail reports, test counts, coverage values, and mutation scores provide useful summaries but omit context needed for acceptance decisions. A reviewer also needs to know which artifact was checked, what behavior the checks covered, where the tests originated, and whether they exposed distinct faults. Recording scope, provenance, adequacy, and diagnostic information makes that context available for subsequent review.

A binary quality gate can conflate the absence of a detected failure with sufficient evidence for acceptance. When checks reveal no violation but leave important requirements unresolved, further verification or human review may be appropriate. We therefore distinguish three outcomes: Accept, Escalate, and Reject.

We propose *Intelligent Testing Infrastructure* (ITI), a reference architecture for coordinating verification and acceptance. ITI brings existing test generation, static and dynamic analysis, and formal verification into a shared workflow: verifiers run under controlled conditions, produce traceable records, and inform versioned acceptance policies. The workflow starts with a submitted candidate artifact; code generation remains outside its core.

An evidence object links each result to the candidate, verifier, scope, adequacy context, provenance, cost, and execution trace. Marginal diagnostic contribution and support breadth describe what a verification portfolio detects beyond its test count. A decision freeze records the evidence and triage outcome before external correctness information enters the evaluation. We distinguish false acceptance rate (FAR), measured among incorrect candidates, from residual accepted-set risk (RAR), measured

among accepted candidates. This distinction matters because accepting fewer incorrect candidates may also exclude correct ones, leaving the accepted set’s error proportion unchanged.

The evaluation draws on four sources with distinct units of analysis. MINI-ITI-POC uses constructed candidates to examine verification, evidence recording, decision freezing, and triage under a composite correctness criterion. EvalPlus supplies public aggregate model–benchmark results, and TC-Bench supplies code–test matrices for diagnostic analysis. A development study runs the complete workflow on 200 model-generated candidates from 50 LiveCodeBench tasks, using model-review proxy labels. The sources address different aspects of ITI and are analyzed separately; neither candidate-level study provides independent correctness adjudication.

The findings show that additional tests can expose failures missed by a basic suite and that diagnostic value depends on complementarity, redundancy, and support structure. They also reveal a limitation of stricter triage: in MINI, the policy lowers FAR without reducing RAR. In the development study, selective escalation reproduces always-on strong-verifier decisions with 34% fewer invocations but shows no improvement in selection under the proxy labels. These results motivate evaluating false acceptance, correct-candidate retention, accepted-set risk, escalation, coverage, and cost together.

1.1 Research Questions

The study is organized around the following research questions.

RQ1: Verification Effectiveness. To what extent can stronger or diagnostically complementary verification reduce the false acceptance of incorrect candidates relative to a basic test suite?

RQ2: Diagnostic Evidence Structure. How do evidence adequacy, complementarity, redundancy, and Support Breadth characterize the diagnostic value of a verification portfolio?

RQ3: Evidence-Conditioned Triage. How can verification evidence support Accept–Escalate–Reject triage, and what trade-offs arise among false acceptance, correct-candidate acceptance, Residual Accepted-Set Risk, escalation, and automated decision coverage?

RQ1 draws on MINI candidate comparisons, EvalPlus aggregate results, and development-study execution outcomes. RQ2 uses MINI cross-detection records and TC-Bench matrices. RQ3 examines MINI’s frozen triage decisions and the development study’s escalation workflow. Development-study correctness metrics are conditional on proxy labels.

1.2 Contributions

This paper makes four contributions with different evidence levels.

1. A reference architecture and operational contract for associating candidate artifacts, verification outcomes, evidence records, and frozen decisions. The proposed invariants concern identity consistency, explicit incompleteness, and the online/offline information boundary; deployment reliability is not established here.
2. An evaluation framework that jointly reports false acceptance, correct-candidate retention, accepted-set error risk, coverage, and cost. Its quantities use established conditional-probability and set-decomposition principles rather than a new probability theorem or a new triage taxonomy.
3. A controlled negative result: task-adequacy triage reduced acceptance without improving accepted-set composition. Separate public-data reanalyses provide context on test strength and detection structure, not additional candidate-level evidence for the triage policy.
4. An end-to-end development study with two model routes showing that selective escalation can reproduce always-on strong-verifier decisions while reducing strong-verifier use. Its proxy-labelled results establish operational feasibility rather than independent correctness discrimination.

The remainder of the paper is organized as follows. Section 2 reviews related work. Sections 3 and 4 describe ITI and acceptance-risk evaluation. Section 5 defines the multi-source empirical design, and Section 6 reports the findings. Sections 7–9 discuss the implications, validity threats, and conclusions.

2 Background and Related Work

2.1 Generated Code and Verification

Early code-generation benchmarks, including HumanEval and MBPP, evaluate function synthesis from natural-language specifications (Chen et al. 2021; Austin et al. 2021). Newer benchmarks and agents address repository patches and multi-step tool use (Jimenez et al. 2024; Yang et al. 2024; Zhang et al. 2024). At this scale, a result can depend on repository state, tools, and the sequence of actions as well as the final code, making execution context part of the artifact under review.

Generated code can pass available tests while violating untested requirements. Code and tests produced by related models may also share assumptions. Evaluations of generated tests therefore examine coverage and fault detection (Wang et al. 2025; Yang et al. 2025; Yu et al. 2025). ITI addresses how verification results are recorded and used to accept a candidate; repository-scale agents are outside the present evaluation.

2.2 Adequacy and Stronger Testing

Adequacy criteria measure a suite against a selected program structure or fault model, whereas an oracle determines how observed behavior is classified (Zhu et al. 1997; Barr et al. 2015). Coverage and test count cannot by themselves establish semantic correctness. Mutation testing estimates sensitivity to selected perturbations, subject to the operators, usable mutants, and the equivalent-mutant problem (Jia and Harman 2011; Papadakis et al. 2019). Property-based and metamorphic testing provide other

ways to derive checks, but their conclusions remain bounded by the encoded properties and relations (Claessen and Hughes 2000; Segura et al. 2016).

EvalPlus demonstrates that strengthened benchmark suites reject some candidates accepted by the original tests (Liu et al. 2023). This finding concerns verifier sensitivity rather than completeness. Security studies likewise show that passing functional tests does not establish secure behavior (Pearce et al. 2022; Perry et al. 2023). Secure-development guidance treats verification and provenance as concerns throughout the software lifecycle (Scarfone et al. 2022; Booth et al. 2024).

2.3 Diagnostic Structure and Provenance

Adequacy assesses a suite against a coverage criterion or fault model; diagnostic diversity concerns whether an additional verifier exposes errors missed by existing checks. TC-Bench represents wrong-code–test relationships as binary matrices, exposing duplicate and zero-detection patterns (Luo et al. 2026). ITI uses these patterns to describe marginal diagnostic contribution and support breadth.

Distinct detection patterns do not prove statistical independence. Diverse software versions can retain correlated failure modes (Littlewood et al. 2001); provenance likewise records origin without proving independent failure behavior. ITI therefore keeps provenance and observed diagnostic structure as separate attributes.

2.4 Selective Decisions and Research Positioning

Continuous integration embeds automated tests in software evolution (Shahin et al. 2017). Selective-classification research supplies the related idea of abstaining when automatic prediction is insufficiently supported (Geifman and El-Yaniv 2017). ITI applies a three-way Accept–Escalate–Reject decision to structured verification evidence, distinguishing an observed violation from an unresolved evidence gap. Supply-chain assurance similarly binds artifacts to verifiable process records, although it addresses a different object and threat model (Torres-Arias et al. 2019; Scarfone et al. 2022).

ITI combines testing, diagnostic analysis, provenance, and selective decisions in a common model for recording verification evidence and deciding whether to accept a candidate. Its contribution lies in this integration and its evaluation across controlled, public-data, and real-model studies, rather than in a new test generator or probability theorem.

3 Intelligent Testing Infrastructure

3.1 Definition, Scope, and Design Principles

ITI is a software-quality reference architecture for coordinating controlled execution, heterogeneous verification, evidence records, and acceptance decisions for versioned candidates. It builds on existing computing, storage, networking, and testing resources. Shared services, orchestration, and persistent records are intended to support reuse and auditability; their reliability in deployment is not evaluated here.

Online verification and recorded decisions

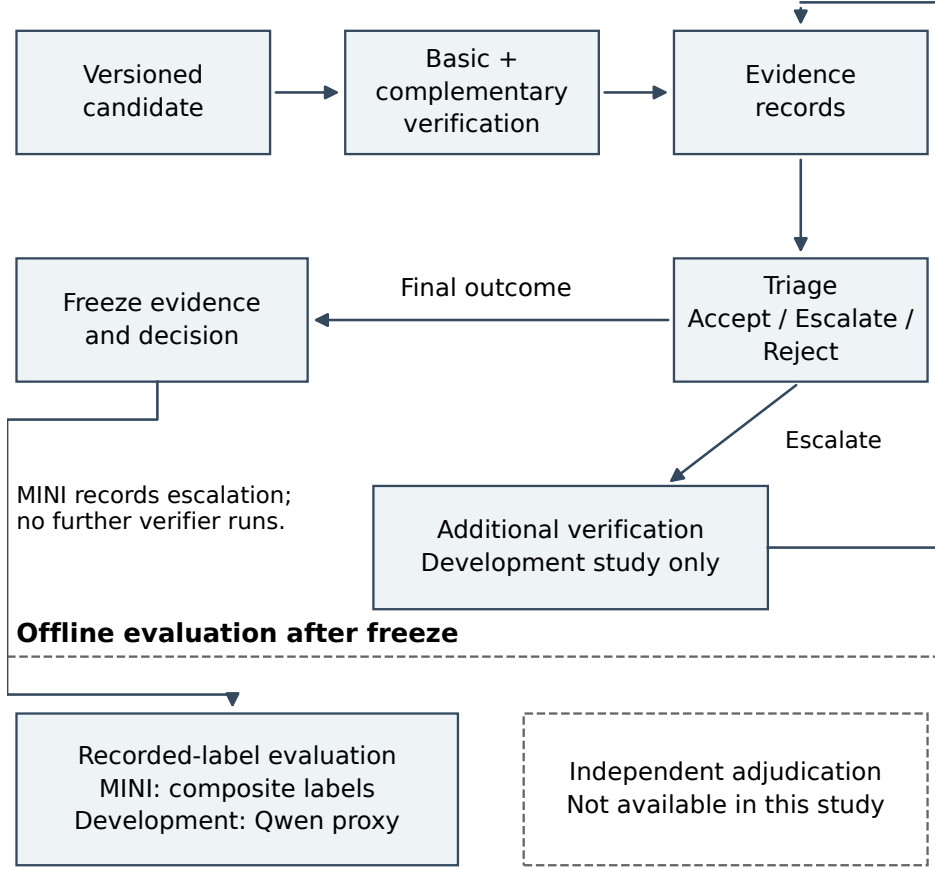


Fig. 1 ITI workflow and evaluation scope. Solid boxes and arrows describe the recorded prototype workflow. The escalation loop is executed in the two-model development study; MINI records escalation without running another verifier. The dashed boundary separates online decisions from subsequent evaluation; it does not establish independent custody. The dashed box denotes independent correctness adjudication required for future confirmation, not a completed experiment. Evaluation uses MINI composite labels and development-study Qwen proxy labels, without revising frozen decisions. Diagram layout and drawing code were prepared with assistance from OpenAI Codex and reviewed by the authors.

Generation and acceptance have separate responsibilities. A candidate may come from an LLM, an agent, a conventional generator, or a human developer. ITI begins when the versioned artifact is submitted for verification, allowing the same recording and decision rules to apply across generators.

The architecture comprises four responsibilities: controlled execution, heterogeneous verification, evidence and ledger management, and evidence-based control. The present study evaluates selected mechanisms within these responsibilities.

3.2 Controlled Execution and Heterogeneous Verification

Controlled execution links an observation to a specified artifact and runtime and limits interference from candidate code. Mechanisms include isolation, resource limits, versioned dependencies and environments, restricted access to evaluation assets, and retained execution traces. These controls support attribution and reproducibility of verification results; they do not establish program correctness.

Tests, static analysis, mutation analysis, property checks, security scanners, and human review contribute observations with different scopes and costs. A verifier adds value when it covers another requirement or exposes failures missed by the current portfolio. Historical MDC and support structure can help assess that value, subject to the evaluated error distribution. Different methods or scopes do not, by themselves, imply independent failures.

3.3 Evidence Organization and Auditability

To make reports interpretable as verification evidence, ITI records artifact identity, verifier and asset versions, outcome, scope, adequacy metadata, provenance, integrity information, cost, and trace. A logical evidence ledger links candidate versions, records, policies, and decisions. Here, *ledger* denotes the record structure, without prescribing a storage technology.

The intended auditability property is the ability to reconstruct an acceptance, escalation, or rejection from the information available at the time. Version binding helps prevent reuse of results from a different candidate or environment. This context extends a test runner’s final pass/fail report.

The operational contract specifies four checks. A result must match the candidate, artifact version, and task identified by its evidence object. Empty suites and infrastructure failures must not count as passes. A confirmed semantic counterexample must remain negative evidence even if other cases fail to execute. Finally, each decision must identify the evidence and policy versions available in that round. These requirements are testable properties of the implementation. Hashes support consistency checks but cannot establish that adjudication data were withheld.

3.4 Online Verification and Offline Evaluation Boundary

The workflow separates online decisions from offline evaluation. Online triage uses only evidence available for the current candidate. Escalation can trigger additional verification and a revised decision before external correctness adjudication is disclosed. The decision freeze then fixes the candidate, evidence, environment, policy, and decision selected for evaluation.

After freezing, hidden tests, independent review, later failures, or other external observations can be linked to the historical decision for evaluation. These observations may inform future policy versions but cannot revise the decision frozen in round k . This constraint governs information flow; it does not ensure that adjudication is complete or independent.

3.5 Evidence-Conditioned Accept–Escalate–Reject Triage

The triage policy distinguishes explicit negative evidence from insufficient positive evidence. Let $V(\mathcal{E}_{on})$ indicate a policy-defined observed violation and let $S_\phi(\mathcal{E}_{on})$ indicate that the online evidence satisfies the acceptance conditions of policy ϕ . The deterministic triage function is

$$C_\phi(\mathcal{E}_{on}) = \begin{cases} \textit{Reject}, & V(\mathcal{E}_{on}) = 1, \\ \textit{Escalate}, & V(\mathcal{E}_{on}) = 0 \wedge S_\phi(\mathcal{E}_{on}) = 0, \\ \textit{Accept}, & V(\mathcal{E}_{on}) = 0 \wedge S_\phi(\mathcal{E}_{on}) = 1. \end{cases} \quad (1)$$

Crucially, the absence of an observed failure does not imply $S_\phi = 1$. Acceptance means only that the current policy’s evidentiary conditions are satisfied. Escalation identifies a need for additional verification or review; it is neither a correctness label nor a completed post-escalation result. MINI instantiates the three states but does not execute a new verifier after escalation.

4 Verification Evidence and Acceptance-Risk Evaluation

4.1 Evidence Objects and Evidence Boundaries

For candidate x and verifier v_i , an Evidence Object is represented as

$$e_i = (o_i, s_i, a_i, p_i, \iota_i, c_i, \tau_i), \quad (2)$$

where o_i is the outcome; s_i is the verified scope; a_i is adequacy metadata; p_i is provenance; ι_i is integrity information; c_i is cost; and τ_i is the execution and version trace. The schema does not require all verifiers to emit identical reports; it requires any result used for acceptance to expose the context necessary to interpret it.

The online evidence set contains only records available to the current decision:

$$\mathcal{E}_{on}(x) = \{e_1, \dots, e_k\}, \quad Y \notin \mathcal{E}_{on}(x), \quad (3)$$

where Y is a post-freeze correctness label. Equation 3 prevents label leakage into the evaluated decision. It does not make a label independent: as Section 5.2 explains, MINI’s final Y is a composite criterion that incorporates Basic and Second passage.

4.2 Evidence Adequacy and Diagnostic Contribution

Evidence adequacy has several dimensions. Coverage, test volume, valid-mutant count, and mutation score describe properties of verification rather than candidate correctness probabilities. We retain adequacy, complementarity, redundancy, provenance, integrity, and support breadth as separate dimensions, without combining them into an uncalibrated score. Provenance and observed adequacy may be available online; measures requiring adjudicated errors belong to offline evaluation.

For an incorrect candidate, let $D_i = 1$ when verifier i detects the error. Given an existing verifier set S , the Marginal Diagnostic Contribution of a new verifier j is

$$MDC(j \mid S) = P(D_j = 1, D_i = 0 \forall i \in S \mid Y = 0). \quad (4)$$

If $U(S)$ denotes the proportion of incorrect candidates detected by at least one verifier in S , then

$$U(S \cup \{j\}) - U(S) = MDC(j \mid S). \quad (5)$$

This equality follows from set decomposition of diagnostic events. Empirical MDC depends on the existing verifier portfolio and the evaluated error distribution.

4.3 Redundancy and Support Breadth

Two tests are observationally redundant on a frozen error set if their detection vectors are identical. A zero-detection test has an all-zero vector. These descriptions apply to the current matrix: repeated tests may serve regression or environmental checks, and a zero-detection test may expose a future error.

Support breadth counts the verification elements that detect a known error. With breadth one, removing or disabling a single test leaves that error undetected by the remaining portfolio. Greater breadth supplies multiple detections but does not establish independent origins or failure mechanisms. Section 5.4 defines its matrix form for TC-Bench.

4.4 Decision Freeze

A decision freeze separates the evidence used for a decision from information later used to calculate MDC, support breadth, FAR, and RAR. The freeze record links candidate hashes, online evidence, policy version, triage outcome, and execution context. Corrections to offline analysis must preserve the historical decision and record the amendment. MINI followed this procedure when correcting its initial label derivation: derived labels and statistics changed, while online evidence and decisions remained fixed.

4.5 False Acceptance and Residual Accepted-Set Risk

False Acceptance Rate and Residual Accepted-Set Risk condition on different populations. FAR asks what proportion of incorrect candidates are accepted; RAR asks what proportion of accepted candidates are incorrect. Correct-candidate acceptance β is required to connect them. If $\pi = P(Y = 0)$ is the error base rate in the evaluation set, then

$$RAR = \frac{\pi FAR}{\pi FAR + (1 - \pi)\beta}, \quad \frac{RAR}{1 - RAR} = \frac{\pi}{1 - \pi} \frac{FAR}{\beta}. \quad (6)$$

The first equality requires a nonempty accepted set; the odds form additionally requires nonzero denominators, including $\beta > 0$. Undefined rates are reported as unavailable rather than zero. Thus, for fixed π , RAR improves only when the policy

reduces acceptance of incorrect candidates relative to acceptance of correct candidates. If FAR and β decline proportionally, RAR is unchanged. When no candidate is accepted, RAR is undefined rather than zero.

4.6 Risk–Coverage–Cost Trade-offs

Triage evaluation therefore requires several measures: FAR, RAR, correct-candidate retention β , escalation rate, terminal-decision coverage, automatic acceptance coverage, and verification or review cost. Escalating almost every candidate can lower FAR while offering little automation or improvement in accepted-set composition.

An appropriate operating point depends on the artifact, quality attribute, consequences of failure, and verification resources. We examine one frozen MINI rule and post-hoc threshold variants; calibrated optimization and deployment risk thresholds are outside this evaluation.

5 Empirical Study Design

5.1 Research Questions and Study Mapping

The four sources address different parts of the research problem. MINI-ITI-POC provides controlled candidate-level records for RQ1–RQ3; EvalPlus provides aggregate model–benchmark results for RQ1; and TC-Bench provides code–test matrices for RQ2. The two-model development study examines the complete workflow and selective escalation for RQ1 and RQ3, using proxy labels. Each source is analyzed with its own units and denominators.

Table 1 summarizes the observations and scope of inference. MINI compares Basic and Basic+Second verification, measures their marginal diagnostic contribution, and evaluates triage under the composite criterion. EvalPlus measures aggregate pass@1 changes after test strengthening. TC-Bench characterizes redundancy, zero detection, and support breadth. The development study evaluates the complete escalation path and reports proxy-conditioned risk and retention.

5.2 MINI Controlled Feasibility Study

MINI-ITI-POC contains 20 Python function tasks spanning five controlled fault families: boundary and critical-value errors; conditional and compound-logic errors; arithmetic, precision, and calculation-rule errors; input-validation and exception-handling errors; and collection, indexing, state, or iteration errors. Each task has four candidate implementations, yielding 80 candidates. The tasks and candidates were constructed to make verification mechanisms observable; they were not sampled from deployed systems or generated by real LLMs. Consequently, the frequency of faults and the proportion of incorrect candidates are design properties rather than estimates of an external population.

Each candidate was evaluated using two content-distinct online test sets. The Basic set represents the initial verification condition, and the Second set provides complementary tests. The two sets support analysis of errors detected by both, by Basic only,

Table 1 Study design and principal inference boundary. The studies are not pooled.

Study	Observations and unit	Adjudication and scope
MINI	20 constructed Python tasks; 80 candidates; task-level mutation context	Composite Basic/Second/hidden label. Controlled mechanism evidence only.
EvalPlus	125 HumanEval/+ and 89 MBPP/+ paired model records	Published aggregate pass@1; no joint candidate labels or triage.
TC-Bench	877 problems; 9,347 selected wrong-code rows; 13,162 test columns	Published detection encodings; diagnostic structure of the frozen object only.
Two-model development	50 LiveCodeBench tasks; 200 Qwen3:8b and DeepSeek candidates	End-to-end development evidence; Qwen review proxy labels, with no independent correctness adjudication.

by Second only, or by neither. They are not claimed to be independent in origin or failure mechanism. A deterministic mutation procedure also evaluated test-suite adequacy at the task level. At most 20 valid mutants were generated per task. A task with fewer than five valid mutants was marked as having weak or insufficient adequacy evidence, and the main triage threshold was fixed at $\theta_{mut} = 0.80$. Mutation adequacy is therefore contextual evidence about a task’s test suite, not a candidate-specific probability of correctness.

The frozen MINI rule rejects a candidate that fails Basic or Second. Candidates passing both suites are escalated if mutation evidence is insufficient or below 0.80, and accepted otherwise. MINI records escalation but runs no subsequent verifier. Its RQ3 analysis therefore evaluates the initial triage rule rather than a completed adaptive workflow.

Decision Freeze fixed candidate and task identifiers, online verification outcomes, task-level mutation evidence, the triage rule, and all triage decisions before hidden evaluation was incorporated. The final correctness variable was then derived as

$$Y(x) = 1 \iff BPass(x) \wedge SPass(x) \wedge HPass(x), \quad (7)$$

where $BPass$, $SPass$, and $HPass$ denote passing Basic, Second, and the hidden test set. Because Equation 7 incorporates the online outcomes being evaluated, Y is a composite criterion rather than an independent correctness label. Freezing precedes disclosure of the hidden component but does not remove this dependence. FAR, RAR, and related counts thus describe controlled reclassification within MINI. The consequences of criterion incorporation are discussed in Section 8.

The historical replication report records re-execution from frozen source in a controlled environment. The pre-adjudication decision file was reproduced byte for byte. Excluding timestamps and runtime fields, core evidence, post-freeze labels, and derived results matched the frozen records. This check supports reproducibility of those records without adding observations or improving sample representativeness.

5.3 EvalPlus Public Benchmark Replay

The EvalPlus analysis uses a fixed snapshot of the public results from the official EvalPlus repository. EvalPlus strengthens the tests associated with HumanEval and MBPP (Chen et al. 2021; Austin et al. 2021; Liu et al. 2023). We extracted paired pass@1 scores for models evaluated with both the original and augmented test suites; we did not generate code, call models, or rerun the benchmarks. The repository revision and the identifier of the selected results file are reported in Table 4.

We include every snapshot record with numeric pass@1 values for both the base and strengthened suites: 125 HumanEval/+ pairs and 89 MBPP/+ pairs, without filtering by model family or performance. An earlier 63/45-record subset could not be reconstructed because its selection CSV was absent from the original package. The present complete-pair analysis replaces that subset; its summaries remain in the historical archive.

For each pair, we subtract strengthened-suite pass@1 from original pass@1 and report the difference in percentage points. The reanalysis includes model identifiers, paired scores, and computed differences. Direction, mean, median, range, and leave-one-record means describe the fixed snapshot. Published score rounding may produce ties. These aggregate records lack the joint candidate outcomes needed to calculate FAR, RAR, or MDC.

5.4 TC-Bench Diagnostic Reanalysis

The diagnostic reanalysis uses a fixed public snapshot of the TC-Bench dataset (Luo et al. 2026). For each problem q , the replay reconstructs a binary matrix A_q in which rows are known wrong-code records and columns are tests; $A_{ij} = 1$ indicates that test j detects wrong code i . The analysis reads the published detection encoding and does not execute the third-party C/C++ submissions. The dataset revision and the content hash of the selected parquet file are reported in Table 4.

Duplicate detection columns have identical vectors within a problem; all but the first occurrence of a vector are counted as duplicates. A zero-detection column contains only zeros for the available wrong-code set. For wrong code w_i , Support Breadth is

$$B(w_i; T_q) = \sum_{j=1}^{|T_q|} A_{ij}. \quad (8)$$

A wrong-code record with $B = 1$ is detected by a single test. This count does not establish test independence. Matrix rank and a deterministic greedy set-cover upper bound provide further descriptions of the observed matrix, rather than semantic independence or correctness guarantees.

The frozen replay comprises 877 problems, 9,347 wrong-code rows, and 13,162 test columns. The upstream source reports 7,016 records described as correct submissions, whereas the frozen parquet contains 6,991 records labelled accepted. The repository does not establish whether these categories are equivalent or why their counts differ; exact cross-source reconciliation therefore remains unresolved. The RQ2 calculations do not use either count, so this discrepancy does not change the reported matrix results, but it remains a data-provenance limitation.

The replay metadata identify the wrong-code rows as a selected diagnostic basis rather than a prevalence sample. Row count equals matrix rank in all 877 problems, so rank observations may reflect dataset construction. Their applicability to other error populations is unknown, and test selection has not been evaluated on held-out wrong programs.

5.5 Metrics and Statistical Analysis

All results are reported first as raw counts and then as percentages where a meaningful denominator exists. For a policy p , False Acceptance Rate, acceptance of composite-labelled correct candidates, and Residual Accepted-Set Risk are

$$\begin{aligned} FAR_p &= \frac{N(\text{Accept}_p \wedge Y = 0)}{N(Y = 0)}, & \beta_p &= \frac{N(\text{Accept}_p \wedge Y = 1)}{N(Y = 1)}, \\ RAR_p &= \frac{N(\text{Accept}_p \wedge Y = 0)}{N(\text{Accept}_p)}. \end{aligned} \tag{9}$$

FAR conditions on incorrect candidates, whereas RAR conditions on the accepted set. A lower FAR therefore does not imply a lower RAR. Escalation rate is the number escalated divided by all candidates, and automated terminal-decision coverage is the number accepted or rejected divided by all candidates.

For MINI, the within-MINI MDC of Second relative to Basic equals the number of composite-labelled incorrect candidates detected by Second but not Basic, divided by all composite-labelled incorrect candidates. Because this label incorporates Second-Pass, the resulting proportion is a descriptive controlled-mechanism quantity rather than an independently adjudicated estimate of diagnostic effectiveness. We also report the complete 2×2 cross-detection counts. Mutation scores are summarized at task level and are not transferred to candidates as correctness measurements. The main triage analysis uses the prespecified threshold 0.80; thresholds 0.60, 0.90, and 1.00 are post-hoc sensitivity settings. A leave-one-task-out analysis checks whether the principal within-sample directions depend on a single constructed task. It is not a confidence interval for a real-task population.

For EvalPlus, each model–benchmark record remains a paired aggregate unit. We report the sign and distribution of the paired pass@1 difference and the range of leave-one-record mean reductions. For TC-Bench, duplicate and zero-detection percentages use test columns as the denominator, while the presence of fragile single-test support uses problems as the denominator. Means and medians of within-problem quantities

weight problems rather than individual wrong-code rows. We do not conduct a cross-study meta-analysis or apply candidate-level inferential tests to aggregate benchmark records.

5.6 Figure preparation

The figures were rendered with Python and Matplotlib. The data plots use the frozen numerical records used by the plotting scripts; the architecture diagram represents the workflow described in Section 3. OpenAI Codex assisted with the plotting code and layout revisions. The authors checked the final figures against the underlying experimental records and reviewed the architecture diagram for consistency with the method.

5.7 Evidence Boundaries and Inference Constraints

Three constraints govern interpretation. First, temporal separation of decisions and evaluation does not make MINI’s composite label independent of Basic and Second. Second, candidate, task, test-column, problem, and model–benchmark metrics retain their respective denominators. Third, observations from the four sources are analyzed separately.

MINI evaluates controlled mechanisms; EvalPlus describes pass@1 changes under stronger tests; TC-Bench characterizes frozen detection matrices; and the two-model study evaluates the workflow and strong-verifier use under proxy labels. These results do not estimate real-world false-acceptance prevalence, calibrated correctness risk, causal benefits of ITI adoption, or industrial effectiveness. Synthetic engineering records are excluded from the scientific analysis.

6 Results

6.1 RQ1 — Verification Effectiveness

MINI’s composite criterion classified 59 of 80 candidates as incorrect and 21 as correct. Basic passed 37 of the 59 incorrect candidates ($FAR_{Basic} = 37/59 = 62.71\%$). Second detected 16 of these candidates, leaving 21/59 (35.59%) passing both visible suites. Because the criterion incorporates both suites, this reduction measures controlled reclassification rather than independently validated correctness discrimination.

In the EvalPlus complete-pair analysis, strengthened tests reduce pass@1 for 122 of 125 HumanEval/+ records; three scores are tied and none increases. The mean reduction is 4.9664 percentage points, the median is 4.90, and the range is 0.0–9.7. All 89 MBPP/+ records decrease, with mean 11.2910 points, median 11.10, and range 2.3–20.4. Leave-one-record mean reductions range from 4.9282 to 5.0065 points for HumanEval/+ and from 11.1875 to 11.3932 for MBPP/+. These statistics were recomputed from the hash-verified snapshot and describe model–benchmark records rather than candidate-level false acceptance.

Figure 2 presents the results with their respective denominators. Both analyses show that additional tests can overturn a pass from a weaker suite. Their effect sizes are not directly comparable because the units and evaluation criteria differ.

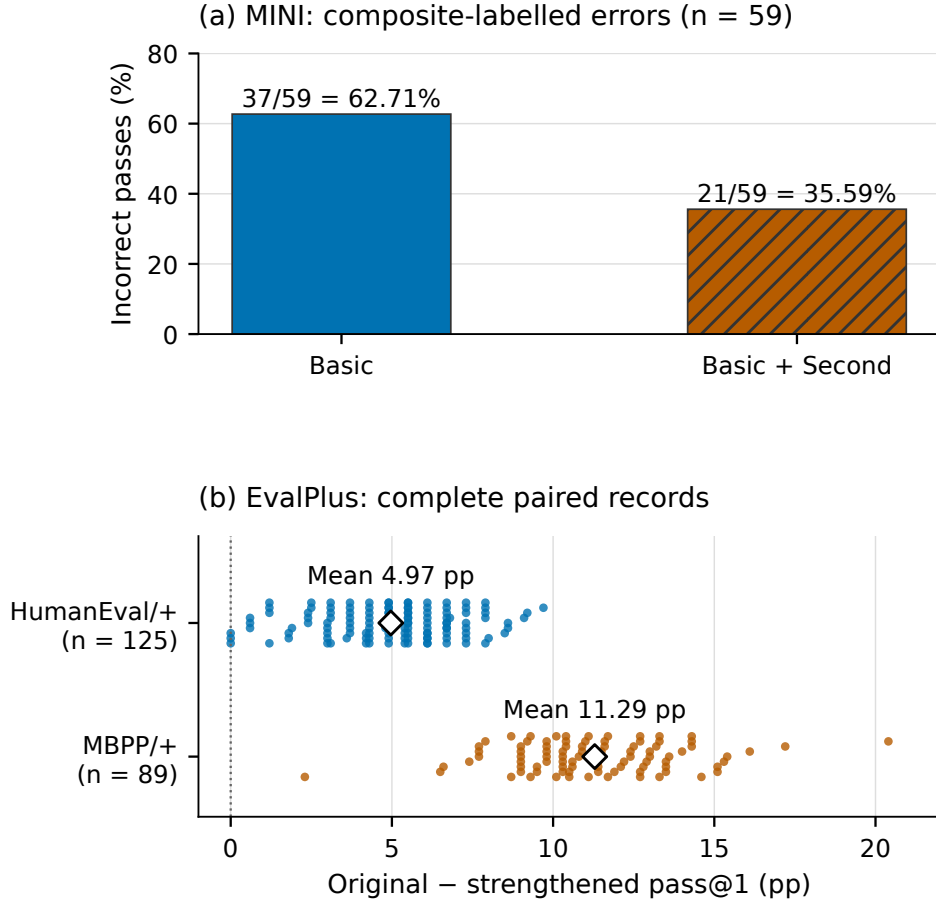


Fig. 2 Effects of stronger tests at two distinct units of analysis. (a) MINI incorrect passes under the composite criterion, with 59 composite-labelled incorrect candidates as denominator. This criterion incorporates Basic and Second outcomes and is not independent adjudication. (b) Published pass@1 reductions for 125 HumanEval/+ and 89 MBPP/+ complete pairs. Each small point represents one model-benchmark record; vertical offsets only separate points. Diamonds mark arithmetic means. Panels do not provide comparable effect sizes.

Answer to RQ1

Within MINI, Second detects 16 candidates missed by Basic, reducing incorrect passes under the composite criterion from 37/59 to 21/59. EvalPlus separately shows lower or equal published pass@1 under strengthened tests, including three HumanEval ties. Additional tests can therefore change observed acceptance; these analyses do not estimate real-world false-acceptance prevalence.

6.2 RQ2 — Diagnostic Evidence Structure

Among MINI’s 59 composite-labelled incorrect candidates, Basic and Second both detected 21, Basic alone detected 1, Second alone detected 16, and neither detected 21. Basic detected 22 in total, Second detected 37, and their union detected 38. Second’s within-MINI MDC is therefore $16/59 = 27.12\%$. This descriptive estimate uses the composite criterion, which incorporates SecondPass. The 21 joint misses also show that positive MDC alone does not establish adequate verification.

Mutation analysis characterized adequacy at the task level. Eighteen tasks yielded numerical mutation scores; 17 of these scored at least 0.80, with mean 0.919 and median 1.0. Separately, eight tasks had fewer than five valid mutants and were marked as providing insufficient adequacy evidence, irrespective of any computable ratio. Some candidates in high-scoring task contexts passed visible tests but failed the composite correctness criterion. Thus, mutation score measures suite sensitivity to the selected operators rather than the correctness of an individual candidate.

The TC-Bench matrices reveal redundancy and fragile support at larger scale. Of 13,162 test columns, 2,966 were duplicate detection columns (22.53%) and 536 were zero-detection columns (4.07%) on the available 9,347 wrong-code rows. At the problem level, 491 of 877 problems contained at least one duplicate column and 235 of 877 contained at least one zero-detection column. The sum of per-problem matrix ranks was 9,347, producing an overall rank-to-test-column ratio of $9,347/13,162 = 0.7102$; 546 problems had more test columns than rank, while 331 had equal values. Rank describes linear matrix structure, not semantic or source independence.

In 725 of 877 TC-Bench problems (82.67%), at least one wrong-code record was detected by exactly one test. This percentage counts problems, not individual errors. Across problems, the mean fraction of wrong-code records with breadth one was approximately 0.2437 and the median was 0.20. A deterministic greedy cover selected a median of 3 tests per problem (mean 3.8757, range 1–16), with a median selected-test ratio of 0.30. This cover applies only to the observed wrong-code rows and provides an upper bound on the minimum cover size; it does not show that three tests establish correctness.

Figure 3 presents the MINI and TC-Bench structures with their distinct denominators. The figure contrasts non-redundant additional detection with duplicate, zero-detection, and fragile-support patterns; it does not combine them into a single evidence-quality score.

Answer to RQ2

Diagnostic value depends on more than suite size. MINI’s Second suite exposes 16 failures missed by Basic, giving a descriptive MDC of $16/59$ under the composite label. TC-Bench reveals repeated detection patterns, zero-detection tests, and frequent single-test support. Test count and mutation adequacy alone do not capture these differences in complementarity, redundancy, and fragility.

6.3 RQ3 — Evidence-Conditioned Triage

At the prespecified mutation threshold of 0.80, MINI accepted 22 of 80 candidates, escalated 20, and rejected 38. The escalation rate was $20/80 = 25.00\%$ and terminal-decision coverage was $(22 + 38)/80 = 75.00\%$. The composite criterion makes every Basic or Second failure incorrect by definition and requires every correct candidate to pass both suites. These properties cannot validate rejection precision or correct-candidate retention independently. Ten of the 20 escalated candidates were composite-labelled correct. No further verification was run, so these counts describe initial triage states.

The RQ3 comparison therefore concerns how mutation context partitions the 42 candidates passing Basic+Second. The comparator accepts all 42, comprising 21 composite-labelled incorrect and 21 correct candidates. Triage accepts 22, with 11 in each group. Consequently, $FAR_{Triage} = 11/59 = 18.64\%$ and $\beta_{Triage} = 11/21 = 52.38\%$, compared with $21/59 = 35.59\%$ and $21/21 = 100.00\%$ for Basic+Second. The comparator’s 100.00% correct-candidate retention follows from the label definition.

RAR was unchanged: Basic+Second accepted 21 incorrect candidates out of 42 ($RAR = 21/42 = 50.00\%$), while triage accepted 11 out of 22 ($RAR = 11/22 = 50.00\%$). Triage reduced acceptance of both composite-labelled groups proportionally and therefore left accepted-set composition unchanged.

A size-matched random-retention comparator makes this limitation concrete. If m of the 42 Basic+Second passes are retained uniformly without replacement, the accepted incorrect count K has the finite-population distribution

$$\begin{aligned} K &\sim \text{Hypergeom}(N = 42, M = 21, n = m), \\ \mathbb{E}[K] &= m/2, \quad \mathbb{E}[RAR] = 1/2 \quad (m > 0). \end{aligned} \tag{10}$$

At $m = 22$, the expected incorrect and correct counts are both 11, matching the observed triage counts. This analytical reference uses the frozen aggregate counts; it is not a randomized experiment or significance test. The operating point shows no composition advantage over size-matched random retention. Testing such an advantage requires candidate-level comparisons that account for task clustering and use independent labels.

Post-hoc threshold analysis reinforced this mixed result. Thresholds 0.60, 0.80, 0.90, and 1.00 produced Accept/Escalate/Reject counts of 22/20/38, 22/20/38, 14/28/38, and 8/34/38, respectively. Across those operating points, FAR decreased from 18.64% to 6.78%, but β decreased from 52.38% to 19.05%, RAR remained 50.00%, escalation increased from 25.00% to 42.50%, and automated terminal-decision coverage decreased from 75.00% to 57.50%. Only 0.80 was fixed before analysis; the other thresholds are sensitivity descriptions, not optimized policies. In 20 leave-one-task-out analyses, the MINI triage-rule FAR ranged from 17.86% to 19.64%, and its RAR remained 50.00% in every case. This check establishes within-MINI stability, not external validity.

Table 2 consolidates the principal observations without combining their units.

Table 2 Principal results with explicit denominators. MINI rates use its composite label. Public reanalyses are separate descriptive units.

Study	Quantity	Value
MINI	Incorrect passing Basic	37/59 (62.71%)
	Incorrect passing Basic+Second	21/59 (35.59%)
	Incorrect accepted by triage	11/59 (18.64%)
	Correct retained by triage	11/21 (52.38%)
	RAR: Basic+Second; triage	21/42; 11/22 (50%)
	Accept / Escalate / Reject	22 / 20 / 38
	Automatic acceptance coverage	22/80 (27.50%)
	Automatic terminal coverage	60/80 (75.00%)
EvalPlus	Mean HumanEval/+ reduction (125 pairs)	4.97 pp
	Mean MBPP/+ reduction (89 pairs)	11.29 pp
TC-Bench	Duplicate test columns	2,966/13,162 (22.53%)
	Zero-detection test columns	536/13,162 (4.07%)
	Problems with single-test support	725/877 (82.67%)

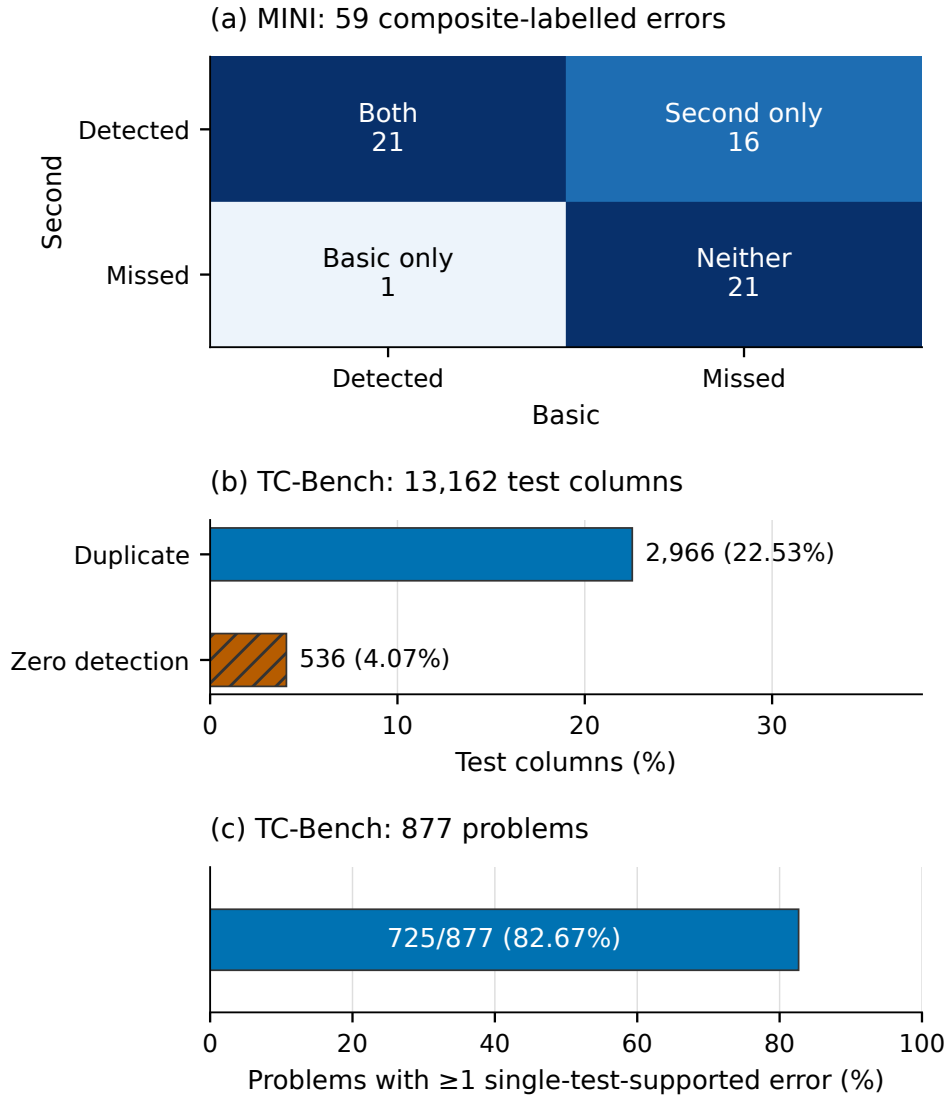


Fig. 3 Diagnostic structure of frozen verification records. (a) MINI cross-detection counts for 59 composite-labelled incorrect candidates; the Second-only cell gives the descriptive MDC of 16/59 (27.12%). (b) Duplicate and zero-detection columns among 13,162 TC-Bench test columns. These properties can overlap. (c) Problems containing at least one wrong-code record detected by exactly one test, using 877 problems as denominator. The panels retain distinct units; neither different detection vectors nor multiple detections establish verifier independence.

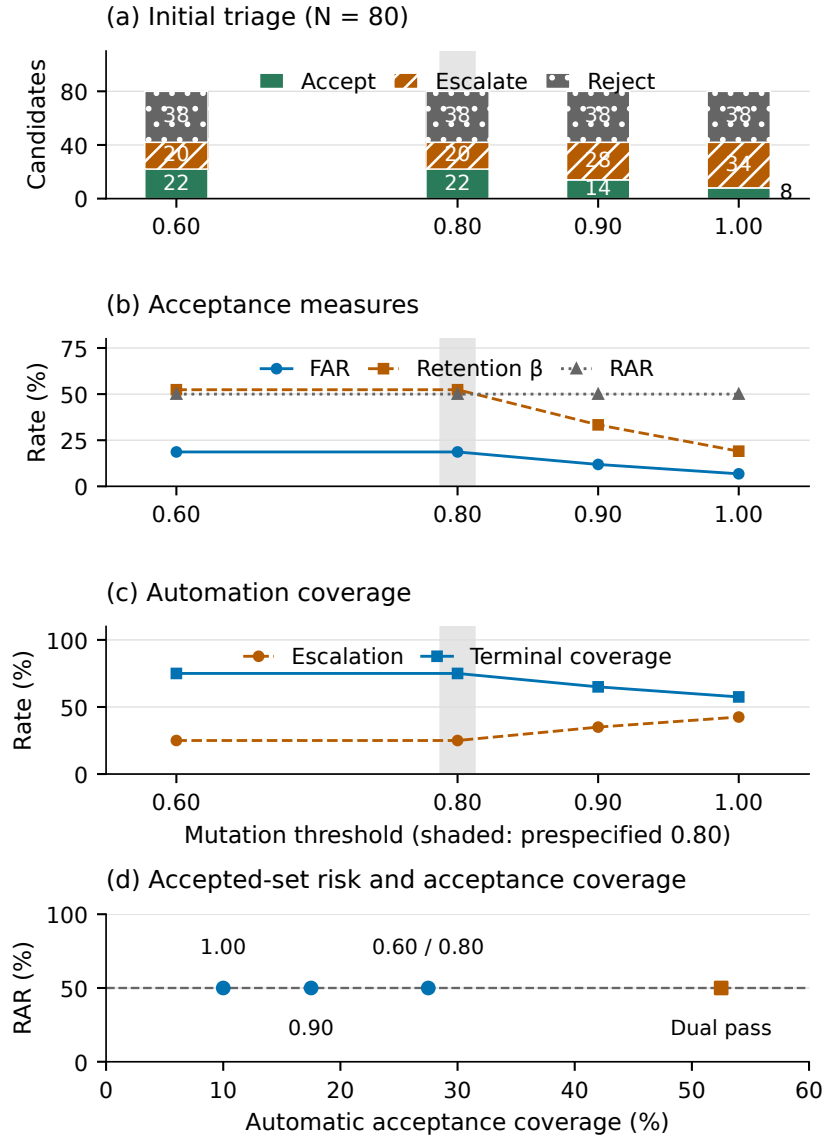


Fig. 4 MINI triage under composite labels. (a) Decisions for 80 candidates. (b) FAR, correct retention β , and RAR. (c) Escalation and terminal coverage. Numeric threshold spacing is preserved; shaded 0.80 was prespecified, and other settings are post-hoc. Lines only connect evaluated settings. (d) Risk versus acceptance coverage. The 0.60/0.80 points coincide. The dashed line is expected RAR under uniform retention from 42 dual-pass candidates, not a fit or confidence band. RAR remains 50%.

Answer to RQ3

MINI demonstrates that verification outcomes and task-level adequacy can define explicit Accept–Escalate–Reject decisions. Compared with accepting all Basic+Second passes, triage lowers FAR but also lowers correct-candidate retention, leaving RAR at 50.00%. The result supports implementation of the triage mechanism, not improved selection or calibrated candidate-level risk.

6.4 Expanded Two-Model Development-Set Study

We executed a development-set study on 50 preselected LiveCodeBench release-v6 tasks. The frozen protocol generated two Python candidates per task from each of two models, for 200 candidates. A local Ollama instance served Qwen3:8b (8.2B parameters, GGUF Q4_K_M) on one RTX 5090 Laptop GPU. The second model was the official DeepSeek API alias `deepseek-flash`; its service fingerprint was retained with each response. Both used temperature 0.6 and a 4,096-token output limit; Qwen used fixed per-candidate seeds, whereas the API did not expose a seed control. All 100 Qwen responses completed. DeepSeek returned 97 complete and three length-truncated responses. No candidate was retried or replaced. Every candidate was evaluated by Basic (V-B), complementary (V-C), and strong (V-E) executable suites in the same restricted WSL2 environment.

Independent benchmark adjudication was unavailable for the complete cohort. Qwen3:8b reviewed each problem and candidate in a separate pass at temperature 0.1. It assigned proxy-correct labels to 45 Qwen and 60 DeepSeek candidates, and proxy-incorrect labels to 55 and 40, respectively. These model judgments are not independent gold labels; Qwen also generated half the cohort. Strong verification yielded 49 passes, 41 failures, and 10 errors for Qwen, and 84 passes, 13 failures, two errors, and one invalid result for DeepSeek. The disagreement between executable outcomes and model judgments leaves correctness unresolved.

Table 3 reports results by model. For Qwen, Basic and Dual accepted 48/100 candidates; All-escalation and ITI accepted 47/100. ITI’s proxy risk was 14/47 (29.79%), with proxy-correct retention of 33/45 (73.33%). For DeepSeek, all five strategies accepted the same 84/100 candidates, yielding proxy risk of 30/84 (35.71%) and retention of 54/60 (90.00%). Across both models, ITI accepted 131/200, with proxy risk of 44/131 (33.59%) and retention of 87/105 (82.86%). A 2,000-replicate percentile bootstrap resampled the 50 tasks, retaining all four candidates from each sampled task. The combined ITI intervals were 23.02–44.80% for proxy risk and 71.43–92.68% for retention. They quantify variation under task resampling, conditional on the proxy labels; they cannot account for errors in those labels.

Five of the 50 task-level mutation contexts were measurable, and four met the frozen adequacy rule; the other 45 contexts were inconclusive. ITI initially accepted nine candidates, rejected 59, and escalated 132. Its final decisions matched All-escalation exactly, with V-E invoked for 132 rather than 200 candidates. Attributed verification time, calculated from the shared execution trace, was 973.81 seconds for ITI and 1,519.64 seconds for All-escalation. These figures describe allocation of observed execution costs rather than independently timed deployments. Generation

Table 3 Two-model development-set study (50 tasks, 200 candidates). Risk and retention use Qwen review proxy labels, not independent correctness. Random retention is the mean of 100 fixed-seed draws from each model’s Dual-pass pool, size-matched to that model’s ITI acceptance count. A/R/U denotes accepted, rejected, and unresolved candidates. For random retention, R denotes candidates not retained, rather than executable failures.

Model	Strategy	A/R/U	Proxy risk	Retention
Qwen	Basic	48/43/9	29.17%	75.56%
	Dual	48/45/7	29.17%	75.56%
	Random retention	47/53/0	29.23%	73.91%
	All-escalation	47/46/7	29.79%	73.33%
	ITI	47/46/7	29.79%	73.33%
DeepSeek	Basic	84/14/2	35.71%	90.00%
	Dual	84/14/2	35.71%	90.00%
	Random retention	84/16/0	35.71%	90.00%
	All-escalation	84/14/2	35.71%	90.00%
	ITI	84/14/2	35.71%	90.00%

took 222.10 seconds for Qwen and 196.41 seconds for DeepSeek; local proxy review took 212.19 seconds.

Figure 5 shows similar Qwen proxy risk and retention for ITI and random retention, and identical DeepSeek acceptance across strategies. The observed benefit is fewer strong-verifier invocations. This study demonstrates the workflow across two model routes without establishing improved correctness discrimination.

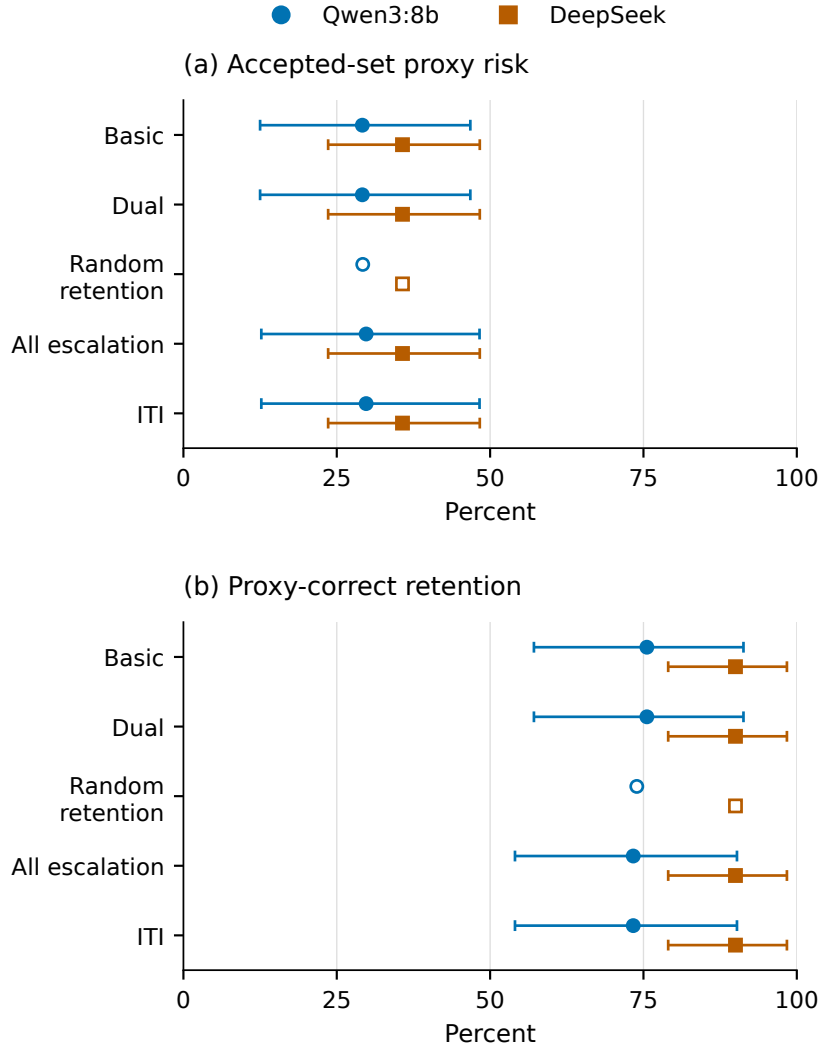


Fig. 5 Two-model development-study results for 50 tasks and 100 candidates per model. Unconnected points show (a) accepted-set proxy risk and (b) proxy-correct retention. Horizontal bars are the original 95% percentile intervals from 2,000 task-cluster bootstrap replicates. Hollow random-retention points are means of 100 fixed-seed, model-stratified, size-matched draws; no bootstrap interval is reported for these means. All metrics use Qwen review proxy labels, not independent correctness adjudication. Marker shape and vertical offsets distinguish the two generators.

7 Discussion

7.1 From Test Passing to Recorded Evidence

The results illustrate how verifier scope affects acceptance. Second rejected 16 MINI candidates that passed Basic, and strengthened EvalPlus suites reduced aggregate pass@1. The interpretation remains conditional: MINI’s criterion includes the evaluated suites, which were constructed within the same study.

An acceptance record should identify the artifact, verifier, assets, environment, scope, adequacy context, and policy. Recording this information makes the decision reviewable and allows later findings to be compared with the evidence originally available.

7.2 Diagnostic Value Beyond Test Count

The RQ2 measures capture different aspects of a portfolio. MDC measures failures exposed by an added verifier and missed by existing ones. Duplicate columns identify repeated patterns, zero-detection columns identify tests detecting none of the represented errors, and support breadth counts tests detecting each error. Combining these measures into a test count or single diversity score would discard those distinctions.

Repeated detection may protect against test or environment failures. On a fixed error set, however, identical detection vectors add execution cost without exposing additional errors. Conversely, a small suite may target important behavior yet remain vulnerable where an error is detected by only one test. Multiple detections and distinct vectors describe observed support, not independent failure behavior.

7.3 Triage, Automation, and Cost

FAR and RAR reveal different consequences of stricter acceptance. MINI’s adequacy rule reduced acceptance of incorrect and correct candidates in the same proportion: FAR fell, but RAR remained 50%. Higher mutation thresholds further reduced automatic coverage without improving composition. Abstention therefore needs to be assessed alongside the quality of the retained set.

The two-model study completed the escalation path omitted from MINI. ITI reproduced All-escalation’s final decisions with 132 rather than 200 strong-verifier calls. The shared trace supports reduced verifier use and attributed cost, although not an independently measured speedup. Selection under the proxy labels showed no advantage over size-matched random retention, and all strategies accepted the same DeepSeek candidates.

7.4 Implications and Next Evaluation

Separating generation from acceptance allows a common verification process to handle candidates from different models. Artifact identities and evidence objects associate results with the submitted code; the decision freeze preserves the evidence and policy

for later review. Error risk, correct retention, escalation, coverage, and cost then provide a fuller basis for choosing an operating point than pass rate alone. ITI organizes these inputs without predicting calibrated correctness probabilities.

A confirmatory study should use independent correctness adjudication, tasks held out from policy development, and prespecified comparators: Basic, Basic+Complementary, All-escalation, budget-matched random escalation, and ITI. Each applicable policy should evaluate the same candidates. Paired comparisons should account for task clustering and report uncertainty for FAR, RAR, correct retention, coverage, and total cost. Ablations should isolate adequacy, complementary verification, and escalation. Online outcomes must be excluded from the gold-label definition, with conflicts between gold labels and online counterexamples resolved through recorded, blinded review. This protocol remains prospective.

8 Threats to Validity

8.1 Construct and Internal Validity

MINI’s correctness criterion combines BasicPass, SecondPass, and HiddenTestPass in Equation 7, so correctness-conditioned metrics partly depend on the outcomes under evaluation. Freezing prevents disclosure of the hidden component during the original decision but does not make the criterion independent. Joint construction of the tasks, candidates, suites, mutation procedure, and triage rule may also influence the results. The post-freeze label correction preserved decisions but underscores the need for independent adjudication.

Mutation adequacy measures task-level sensitivity to selected operators and usable mutants. It does not measure individual candidate correctness. MDC depends on the verifier portfolio and observed error distribution; support breadth and distinct detection vectors likewise describe observed support without establishing independence.

EvalPlus results come from a frozen public aggregate file, and TC-Bench matrices are reconstructed from published encodings. Neither analysis reruns the underlying candidates. Both inherit the source definitions and any upstream execution errors.

8.2 External and Conclusion Validity

MINI contains 80 constructed candidates from 20 Python function tasks. The development set contains 200 candidates from 50 preselected competitive-programming tasks, generated by Qwen3:8b and DeepSeek Flash. Task selection, model versions, Qwen quantization, and WSL2 execution constrain generalization to other models, repository agents, languages, and industrial or security-critical software. Benchmark familiarity or training contamination may also affect the candidate distribution.

The four sources use different statistical units and are not pooled. EvalPlus comparisons are descriptive, and MINI thresholds other than 0.80 are post-hoc sensitivity settings. Development-study intervals resample 50 task clusters; the 100 random-retention draws form one comparator rather than independent experiments. The intervals are wide and remain conditional on Qwen review labels, which may share the generator’s blind spots. Resampling cannot remove this dependence.

8.3 Reproducibility and Evidence Integrity

Historical reports retain revisions and hashes for MINI, EvalPlus, and TC-Bench. MINI’s pre-adjudication decision file was reproduced byte for byte; EvalPlus statistics were recomputed from the fixed snapshot; and TC-Bench summaries were reagggregated from frozen per-problem records. However, the archived package lacks the complete original MINI execution tree and TC-Bench parquet and replay inputs, preventing independent reproduction of every historical result.

An unresolved TC-Bench provenance discrepancy concerns the 7,016 correct submissions reported by the source and the 6,991 accepted records in the frozen parquet. Their equivalence has not been established. RQ2 uses 9,347 wrong-code rows, so these disputed counts do not affect its calculations.

The latest Ubuntu 22.04/WSL2 source passed 99 software tests without skips and 12 scoped sandbox probes. The driver ran on Python 3.12 and isolated candidates on Python 3.10. These checks cover recorded runtime and cleanup behavior, not comprehensive containment. The earlier 95-test record refers to a pre-adapter snapshot. Synthetic engineering runs and the virtual-rater dry run serve only to validate pipelines. The latter derives simulated labels from Qwen proxy labels and strong-verifier outcomes and is excluded from correctness and confirmatory claims.

9 Conclusion

ITI links verification outcomes to candidate artifacts and execution context, distinguishes incomplete evidence from detected violations, and records decisions before offline adjudication. Its evaluation separates false acceptance among incorrect candidates from error risk within the accepted set.

The controlled study yields a negative selection result. MINI’s adequacy-based triage lowers FAR but reduces correct retention proportionally, leaving RAR at 50% and matching the expected composition of size-matched random retention. EvalPlus and TC-Bench separately illustrate sensitivity to test strength and diagnostic structure. In the two-model development study, ITI matches All-escalation’s final decisions on 200 candidates with 132 rather than 200 strong-verifier invocations. Proxy risk and retention show no improvement over the matched random comparator, and all five strategies accept the same DeepSeek candidates.

The findings support further evaluation of ITI as a verification and decision recording architecture. Independent correctness discrimination, calibrated risk, and industrial effectiveness remain unestablished. A confirmatory study with independent adjudication and tasks held out from policy development is needed to determine whether the policy improves selection, beyond reducing acceptance or FAR alone.

Reproducibility Record

Table 4 records the exact upstream objects used in the two public-data reanalyses. A repository revision identifies the state of the source repository or dataset, whereas a blob identifier or SHA-256 digest identifies the selected file content. These identifiers support provenance and integrity checking; they are not experimental outcomes.

Table 4 Upstream source and version record for the public-data reanalyses.

Field	Recorded value
EvalPlus source	Official repository evalplus/evalplus.github.io ; selected object results.json .
EvalPlus version	Repository commit 87418456d3a8c2bc6265f24ff91d2dd0e7993f4b ; file blob SHA-1 21a54c1ea717f1f0d6b66a9a737e76a86d35f0e3 .
EvalPlus inclusion	All records with numeric pass@1 values for both the base and strengthened suite: 125 HumanEval/+ and 89 MBPP/+ pairs.
TC-Bench source	Hugging Face dataset Luoberta/TC-Bench ; selected object default/test/0000.parquet .
TC-Bench version	Dataset revision f4d482da2d015b6342a12b9891149dcb00566c92 ; file SHA-256 9e35749dda4f01239a82eb049d7a29b50681c09a7cfee150983169b57fd687d7 .
TC-Bench inclusion	Published detection encodings for 877 problems, 9,347 wrong-code rows, and 13,162 test columns.

Statements and Declarations

Author contributions

Zhiyuan Li: Methodology, Software, Investigation, Writing – review & editing. **Li Feng:** Conceptualization, Writing – original draft, Writing – review & editing. **Yurui Feng:** Validation, Writing – review & editing. **Chaofan Ji:** Validation, Writing – review & editing. **Jian Xu:** Validation, Writing – review & editing. **Chen Chen:** Validation, Writing – review & editing.

Funding

This research received no specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The EvalPlus, TC-Bench, and LiveCodeBench datasets analysed in this study are publicly available. Table 4 identifies the fixed EvalPlus and TC-Bench snapshots and their file identifiers. The LiveCodeBench release used in the development study is specified in Section 6.4. The study-specific MINI task suite, generated candidates, execution traces, and derived records are not publicly available and will not be supplied on request.

Code availability

The study-specific implementation and analysis code are not publicly available and will not be supplied on request.

Use of generative AI and AI-assisted technologies

During the preparation of this work, the authors used ChatGPT and OpenAI Codex to assist with manuscript organization, English-language editing, and figure preparation code. All authors reviewed and edited the AI-assisted content, checked the references, and verified the reported results and figures against the underlying experimental records. The authors confirm that the final manuscript accurately represents this study and take full responsibility for the content of the published article. The use of language models in the experiments is described separately in the study methods.

References

- Austin J, Odena A, Nye M, et al (2021) Program synthesis with large language models. arXiv preprint arXiv:210807732
- Barr ET, Harman M, McMinn P, et al (2015) The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering* 41(5):507–525. <https://doi.org/10.1109/TSE.2014.2372785>
- Booth H, Souppaya M, Vassilev A, et al (2024) Secure software development practices for generative AI and dual-use foundation models: An SSDF community profile. Tech. Rep. NIST SP 800-218A, National Institute of Standards and Technology, <https://doi.org/10.6028/NIST.SP.800-218A>
- Chen M, Tworek J, Jun H, et al (2021) Evaluating large language models trained on code. arXiv preprint arXiv:210703374
- Claessen K, Hughes J (2000) QuickCheck: A lightweight tool for random testing of haskell programs. In: *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*. Association for Computing Machinery, pp 268–279, <https://doi.org/10.1145/351240.351266>
- Fan A, Gokkaya B, Harman M, et al (2023) Large language models for software engineering: Survey and open problems. In: *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering*. IEEE, pp 31–53, <https://doi.org/10.1109/ICSE-FoSE59343.2023.00008>
- Geifman Y, El-Yaniv R (2017) Selective classification for deep neural networks. In: *Advances in Neural Information Processing Systems*, vol 30. Curran Associates, Inc., pp 4885–4894
- Jia Y, Harman M (2011) An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering* 37(5):649–678. <https://doi.org/10.1109/TSE.2010.62>
- Jimenez CE, Yang J, Wettig A, et al (2024) SWE-bench: Can language models resolve real-world GitHub issues? In: *International Conference on Learning Representations*,

URL <https://openreview.net/forum?id=VTF8yNQM66>

- Littlewood B, Popov P, Strigini L (2001) Modeling software design diversity: A review. *ACM Computing Surveys* 33(2):177–208. <https://doi.org/10.1145/384192.384195>
- Liu J, Xia CS, Wang Y, et al (2023) Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In: *Advances in Neural Information Processing Systems*, vol 36. Neural Information Processing Systems Foundation, pp 21558–21572, <https://doi.org/10.52202/075280-0943>
- Luo X, Huang J, Zheng W, et al (2026) How many code and test cases are enough? evaluating test cases generation from a binary-matrix perspective. In: *International Conference on Learning Representations*, URL <https://openreview.net/forum?id=RomWar2kVN>
- Papadakis M, Kintis M, Zhang J, et al (2019) Mutation testing advances: An analysis and survey. In: *Advances in Computers*, vol 112. Elsevier, p 275–378, <https://doi.org/10.1016/bs.adcom.2018.03.015>
- Pearce H, Ahmad B, Tan B, et al (2022) Asleep at the keyboard? assessing the security of GitHub Copilot’s code contributions. In: *2022 IEEE Symposium on Security and Privacy*. IEEE, pp 754–768, <https://doi.org/10.1109/SP46214.2022.9833571>
- Perry N, Srivastava M, Kumar D, et al (2023) Do users write more insecure code with AI assistants? In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, pp 2785–2799, <https://doi.org/10.1145/3576915.3623157>
- Scarfone K, Souppaya M, Dodson D (2022) Secure software development framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities. Tech. Rep. NIST SP 800-218, National Institute of Standards and Technology, <https://doi.org/10.6028/NIST.SP.800-218>
- Segura S, Fraser G, Sanchez AB, et al (2016) A survey on metamorphic testing. *IEEE Transactions on Software Engineering* 42(9):805–824. <https://doi.org/10.1109/TSE.2016.2532875>
- Shahin M, Babar MA, Zhu L (2017) Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access* 5:3909–3943. <https://doi.org/10.1109/ACCESS.2017.2685629>
- Torres-Arias S, Afzali H, Kuppusamy TK, et al (2019) in-toto: Providing farm-to-table guarantees for bits and bytes. In: *28th USENIX Security Symposium*. USENIX Association, pp 1393–1410

- Wang W, Yang C, Wang Z, et al (2025) TestEval: Benchmarking large language models for test case generation. In: Findings of the Association for Computational Linguistics: NAACL 2025. Association for Computational Linguistics, pp 3547–3562, <https://doi.org/10.18653/v1/2025.findings-naacl.197>
- Yang J, Jimenez CE, Wettig A, et al (2024) SWE-agent: Agent–computer interfaces enable automated software engineering. In: Advances in Neural Information Processing Systems, vol 37. Neural Information Processing Systems Foundation, pp 50528–50652, <https://doi.org/10.52202/079017-1601>
- Yang Z, Kuang Z, Xia X, et al (2025) Can LLMs generate high-quality test cases for algorithm problems? TestCase-Eval: A systematic evaluation of fault coverage and exposure. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Association for Computational Linguistics, pp 1050–1063, <https://doi.org/10.18653/v1/2025.acl-short.82>
- Yu B, Zhu Y, He P, et al (2025) UTBoost: Rigorous evaluation of coding agents on SWE-bench. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, pp 3762–3774, <https://doi.org/10.18653/v1/2025.acl-long.189>
- Zhang Y, Ruan H, Fan Z, et al (2024) AutoCodeRover: Autonomous program improvement. In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis. Association for Computing Machinery, pp 1592–1604, <https://doi.org/10.1145/3650212.3680384>
- Zhu H, Hall PAV, May JHR (1997) Software unit test coverage and adequacy. ACM Computing Surveys 29(4):366–427. <https://doi.org/10.1145/267580.267590>