BETTER RISK HYPOTHESES, BETTER CODE REVIEWS? DIAGNOSING RISK FORMULATION AND PRIORITIZATION IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

AI-assisted coding can accelerate code production, increasing the need for efficient and reliable code review. Yet evaluating LLM reviewers through final findings alone obscures whether failures arise from overlooking relevant risks, prioritizing unproductive investigations, or failing to verify a concern. We introduce a diagnostic framework centered on Risk Hypotheses: structured, testable descriptions of potential failures and the evidence needed to investigate them. From real review histories, we curate 300 reference risks grounded in reconstructed review-time evidence, distinguishing concerns worth investigating from those verifiable in a fixed environment. The framework evaluates risk formulation, budgeted prioritization, and risk-guided review. Controlled comparisons isolate selection effects using fixed candidate pools and verifiers, while Gold-derived guidance and field ablations probe how hypothesis content affects downstream findings. Initial experiments show that broader risk coverage does not necessarily improve top-ranked coverage, while Gold-derived guidance can improve finding recall for some reviewers. We also identify grounded risks that require evidence unavailable in the review environment. Together, these observations show why recognizing a relevant risk, prioritizing its investigation, and substantiating a finding require separate assessment. Our framework connects these intermediate capabilities to review outcomes, enabling systematic diagnosis of when better risk hypotheses lead to better code reviews.

1 INTRODUCTION

Effective code review depends on deciding what to investigate. Given a code change, a reviewer must identify potential problems, choose which ones warrant attention, and gather evidence from the surrounding repository. These decisions become consequential when time and context are limited: a relevant concern may receive too little attention, while a plausible but unproductive investigation consumes the available budget. Final review comments reveal what a model reports, but offer limited insight into how its investigation led to that outcome.

Recent work has expanded code-review evaluation to include repository-level context (Zhang et al., 2026b), structured review checkpoints (Zhang et al., 2026a), and the balance between useful findings and false alarms (Pereira et al., 2026). In vulnerability detection, explicit hypotheses also guide the investigation and validation of potential issues (Wang et al., 2026). These developments motivate examining the concerns that precede a review finding. When an issue is missed, can we distinguish a risk that was never proposed from one that was proposed but not selected, or selected but not substantiated?

Consider a patch that adds a scheduled task writing a marker to a local file, together with a test that checks for the marker from another process. A comment describing the test as "potentially flaky" leaves the cause unclear. A more informative hypothesis states that, if the processes use isolated filesystems, the test may fail to observe the marker even when the task succeeds. This hypothesis identifies both an unresolved condition and a concrete investigation: checking whether the processes share the relevant storage.

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.

We formalize such concerns as *risk hypotheses*: repository-grounded descriptions of possible failures and the evidence needed to investigate them. A hypothesis can be worth investigating before the available evidence establishes a reportable issue. Our *Risk Hypothesis Benchmark* contains 300 reference risks across 223 review cases from 157 repositories, constructed from historical reviews and reconstructed review-time code. These references support evaluation of risk formulation, while a subset with feasible verification paths supports evaluation of downstream findings.

The benchmark enables four complementary comparisons. RQ1 examines whether explicit hypothesis generation changes risk coverage and ordering relative to ordinary review generation. RQ2 holds the candidate pool and verification budget fixed and varies the selected hypotheses. RQ3 compares direct review with a workflow that generates and investigates its own hypotheses under a matched total budget. RQ4 removes individual fields from reference-derived guidance to examine which information contributes to verification.

The results suggest that relevant risk coverage and effective review need separate assessment. Explicit hypothesis generation has higher observed Top-10 coverage than normalized review comments in four paired model comparisons. For DeepSeek, however, most additional matches occur at ranks 6–10, with no consistent improvement at rank one. Oracle selection increases finding recall from a fixed pool, whereas self-generated guidance yields lower observed recall than direct review for both tested backbones under matched total budgets. Reference-derived guidance attains higher recall in a separate descriptive comparison, and removing the verification plan or failure mechanism produces the largest observed drops in the field ablations. Together, these observations suggest that useful risk information can assist investigation, while generating and using that information within a limited budget remains a distinct challenge.

Our contributions are:

1. A benchmark of repository-grounded risk hypotheses that separates concerns worth investigating from issues verifiable under a specified evidence interface.
2. An empirical evaluation of risk formulation, budgeted selection, and verification, with controlled selection comparisons, matched-budget review workflows, and guidance-field ablations.

2 Problem Formulation

2.1 Review instance and available evidence

A review instance i consists of a repository snapshot S_i, a code change Δ_i, and task context X_i, such as the PR description. The reviewer examines the change and its surrounding code under a resource budget B_i, producing a set of findings F_i.

Let E_i denote the review-time evidence accessible through the permitted actions $\mathcal{A}_i$, such as source inspection and, where supported, test execution. The reviewer may acquire only part of this evidence during its investigation.

Historical comments and later fixes support reference construction. The evaluated reviewer receives the reconstructed review-time context, with these annotation sources withheld.

2.2 Risk hypotheses and findings

A risk hypothesis describes a possible failure and how to investigate it. We represent it as

$$h = (a, t, m, o, e, v, s), \tag{1}$$

where a is the affected entity, t the trigger, m the failure mechanism, o the conditional impact, e the evidence needed, v the verification plan, and s a summary statement. Together, these fields connect a specific program relationship to a testable concern.

Investigation may support, refute, or leave a hypothesis unresolved. A finding is a reported issue accompanied by supporting evidence. Its validity depends on whether that evidence substantiates the claim. We evaluate findings along two dimensions: validity, which assesses whether the issue is supported, and reference matching, which assesses whether it identifies a concern represented in

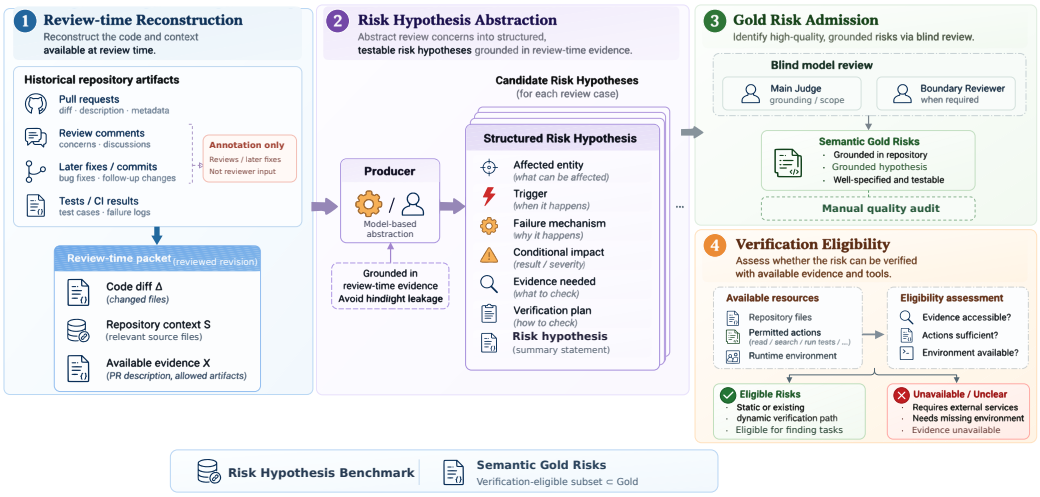

Figure 1: Overview of benchmark construction: review-time reconstruction, risk abstraction, Gold admission, manual quality auditing, and verification eligibility assessment.

the Gold set. This allows valid findings beyond the reference set to be recognized separately from reference coverage.

Let $\mathcal{G}_i$ denote the reference risk hypotheses for instance i. For downstream task q, the eligible subset is

$$\mathcal{G}_i^{(q)} = \{g \in \mathcal{G}_i : \text{eligible}(g, E_i, \mathcal{A}_i, q) = 1\}. \tag{2}$$

A hypothesis is eligible when the available evidence and actions support the investigation required by the task. Formulation is evaluated against $\mathcal{G}_i$; downstream finding recall measures coverage of $\mathcal{G}_i^{(q)}$ by valid, matched findings. The matching rubric specifies the issue condition associated with each reference risk.

2.3 Formulation, Selection, and Verification

For instance i, formulation produces an ordered candidate pool H_i. A selector passes at most K hypotheses to a verifier, which investigates them under budget B_i^{ver} and may also report other findings supported by repository evidence.

3 Benchmark Construction

We construct the benchmark from 33,596 pull-request records in the curated AIDev subset (Li et al., 2026). The construction pipeline comprises LLM-based candidate screening, risk abstraction, review-time evidence verification, model-based Gold admission, and review by members of the research team. The resulting benchmark contains 300 Gold Risks across 223 review cases from 157 repositories. Figure 1 summarizes the process.

3.1 Source Selection and Review-time Reconstruction

We use LLM-based screening to identify candidate concerns from reviews of agent-authored pull requests in the frozen AIDev subset. For each candidate, we identify the reviewed revision using PR and review metadata together with commit ancestry. We reconstruct its diff, changed-file list, and relevant source context to form the review-time evidence packet. Historical review comments support reference construction and are withheld from the evaluated reviewer.

3.2 CANONICAL REPRESENTATION

Each Gold Risk uses the seven fields in Table 1. Its identifiers, repository and PR, snapshot, source provenance, construction version, and admission status are stored as metadata. We do not assign a priority score; RQ2 evaluates selection through downstream findings.

Table 1: The seven semantic fields of a Gold Risk. Provenance and task eligibility are stored as metadata.

Field	Description
`affected_entity`	The function, component, artifact, or program relationship affected by the concern.
`trigger`	The change and conditions under which the concern arises.
`failure_mechanism`	The causal relationship that could produce a failure, with unresolved premises stated explicitly.
`potential_impact`	The bounded consequence if the trigger and mechanism hold.
`evidence_needed`	Evidence that would support or refute an unresolved premise.
`verification_plan`	An investigation beginning with a check that distinguishes the competing possibilities.
`risk_hypothesis`	A concise summary consistent with the other six fields.

3.3 RISK ABSTRACTION

Historical review comments identify candidate concerns but may be vague (Liu et al., 2025). A model-based Producer uses the review discussion, later fixes, and tests to interpret each concern, then expresses it as a hypothesis about the reviewed revision. The hypothesis must be grounded in review-time code. Plausible but unresolved premises must be stated explicitly and paired with a check; unsupported speculation is excluded. A discussion may be split, merged with related evidence, or partially retained when the resulting hypothesis remains traceable to its source.

3.4 MODEL-BASED GOLD ADMISSION

The Main Judge assesses each hypothesis against its review-time evidence packet without seeing historical comments, later fixes, observed outcomes, or researcher explanations. It checks grounding, scope, and treatment of unresolved premises. External-contract assumptions and specified disagreements receive Boundary Reviewer or additional reference review. Candidates passing these checks proceed to manual quality review before inclusion in the final Gold set.

3.5 MANUAL QUALITY REVIEW

Members of the research team reviewed the model-admitted candidates against their reconstructed review-time evidence. The review assessed code grounding, the treatment of unresolved premises, consistency across the seven fields, the usefulness of the verification plan, and potential hindsight leakage. This review formed the final quality check for the 300 Gold Risks used in our experiments.

3.6 DOWNSTREAM VERIFICATION ELIGIBILITY

For finding-level tasks, we assess whether the permitted evidence and actions can test each risk's target, trigger, mechanism, potential impact, and counterevidence. We record an investigation path and the conditions for matching a finding, and classify verification as `STATIC`, `EXISTING_DYNAMIC`, or `UNAVAILABLE`, accounting for external contracts, generated artifacts, and execution dependencies. Risks without a feasible path remain formulation targets but are excluded from the corresponding finding evaluation. Eligibility is withheld from reviewer inputs, and evaluation cohorts are fixed before observing downstream outcomes.

4 EVALUATION DESIGN

We examine four questions: can models formulate and order relevant risks (RQ1); does better selection improve verification (RQ2); does explicit planning improve end-to-end review (RQ3); and which guidance content helps (RQ4)? RQ2 uses a fixed Flash-generated candidate pool with each verifier, whereas RQ3 uses the same model backbone for risk generation and verification.

4.1 RQ1: RISK FORMULATION AND RANKING

DIRECT REVIEW (DR) generates up to ten ordinary review comments. REVIEW-NORM converts those comments one-to-one into the risk schema, preserving their order and concerns. DIRECT HYPOTHESIS (DH) independently generates up to ten seven-field risks from the same review-time packet. The normalization control accounts for presentation differences, while the comparison against raw DR checks sensitivity to normalization.

We evaluate DeepSeek Flash, DeepSeek V4 Pro, GLM, and Qwen Max, using a separate jointly evaluable paired set for each model.

Broad matching identifies the same overall concern. Strict matching additionally requires alignment of the affected target, trigger, and failure mechanism. For matching level ℓ,

$$\mathrm{Recall}_\ell @K = \frac{\sum_i \sum_{g \in \mathcal{G}_i} \mathbb{1}[\exists h \in H_i^{\le K} : M_\ell(h, g) = 1]}{\sum_i |\mathcal{G}_i|}. \tag{3}$$

We report coverage at $K \in \{1, 3, 5, 10\}$, candidate counts, and output availability. MRR and NDCG provide additional descriptions of candidate ordering. These metrics concern reference relevance rather than defect severity.

The RQ1 analysis uses paired cases with observable outputs and complete matching for both conditions. Legal empty outputs count as zero; invalid or unevaluated outputs are excluded. We compute 95% confidence intervals from 10,000 paired bootstrap resamples clustered by PR.

4.2 RQ2: BUDGETED RISK SELECTION

RQ2 tests whether selecting more relevant risks improves finding recall under a fixed verification budget. We use a common pool of seven-field hypotheses generated by DeepSeek Flash in RQ1. Each selection strategy chooses up to three hypotheses per PR.

ORIGINAL selects the first three hypotheses in generation order. ORACLE SELECTION uses candidate-to-Gold matching labels to prioritize coverage of distinct eligible Gold risks. Both strategies select existing candidates from the same pool; their content remains unchanged.

We additionally evaluate three selection baselines with the DeepSeek Pro verifier. RANDOM selects candidates using a fixed random ordering. BM25 ranks candidate hypotheses by lexical relevance to the PR and diff context. GENERIC LLM selects hypotheses from the same pool using the review-time context. These three strategies have no access to Gold risks or candidate-to-Gold matching labels and do not modify candidate content.

The selected hypotheses are presented in a canonical order, with selection scores and strategy identities removed. Repository context, tools, verifier prompts, and verification budgets are held fixed within each comparison. We compare Original and Oracle Selection separately with GPT-5.6-sol and DeepSeek Pro as verifiers, using the same Flash-generated candidate pool. The additional selection baselines are compared with Original on jointly evaluable PRs in the Pro setting.

We measure coverage before and after selection to locate losses between risk generation and verification. Let C_i denote the eligible Gold risks covered by the full candidate pool and T_i those covered by the selected subset. Pool coverage is $|C_i|/|\mathcal{G}_i^{(2)}|$, and selection retention is $|T_i|/|C_i|$ for PRs with nonempty C_i. Final findings are evaluated against the full eligible reference set, including findings beyond the selected guidance. The primary comparison is the paired difference in PR-level Strict Finding Recall between Oracle Selection and Original.

4.3 RQ3: RISK-GUIDED REVIEW

RQ3 tests whether explicitly formulating risks before investigation improves end-to-end code review. DIRECT investigates the change and reports findings. SELF first generates risk hypotheses using the seven-field representation, selects up to three in generation order, and passes them to a separate verification stage. Planning and verification use the same model backbone. We compare the two workflows separately for DeepSeek Flash and Pro on 60 PRs containing 68 eligible Gold risks.

Both workflows use the same repository snapshots and evidence-acquisition tools. Their total resource allocations are matched:

$$B_{\text{Direct}} = B, \qquad B^{\text{plan}}_{\text{Self}} + B^{\text{ver}}_{\text{Self}} = B. \tag{4}$$

Here, B specifies limits on tokens, tool calls, tool-return volume, and elapsed time. Self divides this allocation between planning and verification, while Direct uses it for review.

The main comparison reports PR-macro Strict Finding Recall over the full 60-PR cohort and the Self-minus-Direct difference. We also examine paired differences on jointly evaluable PRs. Finding evidence must be traceable to the review trajectory.

We additionally evaluate GOLD GUIDANCE as a separate diagnostic with the Pro verifier on the same cohort. This condition supplies reference-derived risk hypotheses directly to verification, using the same verification-stage budget as Self without running the planning stage. Its comparison with Direct and Self is descriptive and does not constitute an equal-cost end-to-end comparison.

4.4 RQ4: INFORMATION SUPPLIED IN RISK GUIDANCE

RQ4 examines which information in a risk hypothesis helps the reviewer identify valid findings. We vary the guidance supplied to the same verifier while holding repository context, tools, and verification budget fixed. Guidance is derived from reference risks to isolate its content from errors in candidate generation and selection.

We compare eight conditions. FULL provides all seven fields: affected entity, trigger, failure mechanism, potential impact, evidence needed, verification plan, and risk hypothesis. Each of the other seven conditions removes one field while leaving the remaining fields unchanged. Comparing each variant with Full measures the additional value of the removed field when the rest of the guidance is still available.

The experiment uses DeepSeek V4 Flash on 30 development PRs containing 35 eligible reference risks. We compare Strict Finding Recall and finding validity precision across conditions, using paired PR-level differences in Strict Finding Recall on evaluable results to assess each field's contribution.

4.5 FINDING EVALUATION AND STATISTICAL ANALYSIS

Text similarity alone is a poor proxy for review quality (Lu et al., 2024). CRScore evaluates comments against code-derived claims and potential issues (Naik et al., 2025), while HalluJudge assesses whether comments are grounded in the supplied code context (Tantithamthavorn et al., 2026). These approaches motivate evaluating the substance of a finding separately from its overlap with a reference.

We evaluate findings in two stages. First, a validity judge assesses whether each finding is supported by repository evidence. Valid findings are then matched to eligible reference issue conditions using the strict matching rubric. For the controlled comparisons, condition names are withheld from both evaluation stages. The separate Gold Guidance diagnostic uses a non-blinded retrospective evaluation, so its comparison with the existing Direct and Self evaluations is interpreted descriptively.

For PR i under condition c, Strict Finding Recall is

$$R_{i,c} = \frac{\sum_{g \in \mathcal{G}_i^{(q)}} \mathbb{1}[\exists f \in F_{i,c} : \text{Valid}(f) \wedge M_F(f, g)]}{|\mathcal{G}_i^{(q)}|}, \tag{5}$$

where $M_F(f, g)$ indicates a strict match between finding f and reference condition g. Each reference contributes at most one hit, regardless of how many findings match it.

We report PR-macro recall, which weights PRs equally, and Gold-micro recall, which aggregates matched references across PRs. Validity precision is the proportion of findings judged valid among those with completed validity assessments. Valid findings outside the reference set contribute to precision but not reference recall.

Comparisons use paired PR-level recall differences, win/tie/loss counts, and paired-bootstrap confidence intervals where available. We report the number of PRs included in each comparison. Analyses of jointly evaluable PRs require measurable outcomes for both conditions. Full-cohort analyses additionally assign zero recall to runs without an evaluable outcome, measuring the findings delivered across all assigned PRs. Completed empty outputs receive zero recall; precision is undefined when no findings are returned.

5 RESULTS

5.1 RQ1: RISK COVERAGE AND RANKING

Direct Hypothesis (DH) achieves higher Strict Recall@10 than Review-Norm in all four model comparisons on jointly evaluable cases (Table 2). The gains range from 7.80 to 12.56 percentage points, but improvements at the top of the ranking are less consistent.

Table 2: RQ1 Gold-micro Strict Recall@10 (%). Each model uses its own jointly evaluable paired set. Δ is DH minus Review-Norm, computed before rounding.

Model	Review-Norm	DH	Δ (pp)
DeepSeek Flash	51.38	59.17	+7.80
DeepSeek V4 Pro	35.87	48.43	+12.56
GLM	32.1	41.5	+9.4
Qwen Max	45.9	55.7	+9.8

For DeepSeek Flash and Pro, most of the additional coverage appears at ranks 6–10, alongside longer candidate lists. Strict Recall@1 changes by +2.29 points for Flash and -1.35 points for Pro. Qwen Max improves at both ends of the list: Recall@1 rises from 8.2% to 16.4%, and Recall@10 from 45.9% to 55.7%. GLM improves over Review-Norm at rank ten, although the comparison with raw Direct Review reverses direction. Its result therefore depends on the baseline representation and the paired evaluation subset.

Explicit formulation expands risk coverage in these normalized comparisons, but the additional relevant risks do not consistently receive the highest priority.

5.2 RQ2: SELECTING RISKS FOR VERIFICATION

Oracle Selection achieves higher PR-macro Strict Finding Recall than Original with both verifiers on jointly evaluable PRs (Table 3). Recall rises from 50.00% to 68.33% with DeepSeek V4 Pro and from 50.94% to 63.21% with GPT-5.6-sol. Both comparisons use the same candidate pool and verification budget; only the selected candidates change.

Table 3: RQ2 PR-macro Strict Finding Recall (%). Each row uses jointly evaluable PRs for that comparison, so Original values vary across rows. Δ is Selected minus Original, computed before rounding.

Verifier	Selection	Original	Selected	Δ (pp)
GPT-5.6-sol	Oracle	50.94	63.21	+12.26
DeepSeek V4 Pro	Oracle	50.00	68.33	+18.33
	Random	52.00	36.00	-16.00
	BM25	36.00	40.00	+4.00
	Generic LLM	45.00	50.00	+5.00

Candidate coverage helps explain the Oracle result. On PRs with complete candidate-to-Gold labels, Original's Top-3 covers 53.03% of eligible Gold risks, compared with 71.21% for Oracle. Among Gold risks already represented in the pool, Original retains 74.47%, whereas Oracle retains 100%. Oracle therefore shows the available improvement from selecting relevant candidates within the existing pool; it uses Gold information and is not a deployable selector.

The practical selectors show smaller and more variable changes. With DeepSeek V4 Pro, BM25 and Generic LLM increase recall by 4.00 and 5.00 points on their respective paired subsets, while Random lowers it by 16.00 points. Generic LLM also improves Top-3 Gold coverage by 4.62 points on the common label-complete subset. However, approximately one quarter of label-complete PRs have no candidate that strictly matches an eligible Gold risk. Selection can recover relevant candidates overlooked by Original, but cannot supply risks absent from the pool.

5.3 RQ3: Risk guidance and review outcomes

Under the matched end-to-end budget, Self has lower PR-macro Strict Finding Recall than Direct for both backbones (Table 4). Recall decreases from 21.67% to 11.67% for Flash and from 22.50% to 11.67% for Pro. Gold-micro recall follows the same direction, decreasing from 19.12% to 10.29% and from 20.59% to 10.29%, respectively.

Table 4: RQ3 review outcomes (%). Macro R and Micro R denote PR-macro and Gold-micro Strict Finding Recall. Prec. is reported finding validity precision; PR hit and Valid PR are the proportions of PRs with a strict Gold match and a valid finding, respectively. Recall and PR-level rates use the fixed evaluation cohort; precision uses judged findings. Self and Gold Guidance retain the original trajectory-based admission rule. † marks separate non-blind, post-hoc evaluation.

Condition	Macro R	Micro R	Prec.	PR hit	Valid PR
DeepSeek Flash					
Direct	21.67	19.12	41.94	21.67	30.00
Self	11.67	10.29	39.22	11.67	21.67
Gold Guidance†	34.17	32.35	66.67	35.00	36.67
DeepSeek V4 Pro					
Direct	22.50	20.59	73.68	23.33	38.33
Self	11.67	10.29	90.48	11.67	25.00
Gold Guidance†	41.67	41.18	81.58	43.33	46.67

The precision results differ across backbones. Flash Self and Direct have similar validity precision, at 39.22% and 41.94%. Pro Self has higher reported precision than Direct, at 90.48% versus 73.68%, despite its lower recall. The proportion of PRs with a strict Gold match also declines under Self, from 21.67% to 11.67% for Flash and from 23.33% to 11.67% for Pro. Generating explicit risks before verification therefore does not by itself improve review coverage under the shared budget.

Reference-derived Gold Guidance has higher reported recall: 34.17% for Flash and 41.67% for Pro, with validity precision of 66.67% and 81.58%. Strict Gold matches occur on 35.00% and 43.33% of PRs. Gold Guidance uses Self's verification-stage budget, excluding risk generation, and is assessed separately without blinding. These are exploratory comparisons and cannot isolate the effect of guidance from differences in evaluation. A separate snapshot-content sensitivity analysis gives Gold Guidance macro recall of 39.17% for Flash and 43.33% for Pro; these values are not mixed into the main comparison.

5.4 RQ4: Contributions of individual risk fields

Table 5 compares full seven-field Gold Risk guidance with single-field deletions using DeepSeek V4 Flash. Each comparison uses jointly evaluable PRs and leaves the remaining fields unchanged. This exploratory ablation uses development and previously exposed PRs.

Removing the verification plan gives the largest observed recall drop, from 65.5% to 50.0%. Removing the failure mechanism lowers recall from 65.5% to 55.2%. These point estimates suggest that guidance about how a failure may arise and how to investigate it is useful to the verifier.

Table 5: RQ4 paired PR-macro Strict Finding Recall (%). Δ is Full minus Ablated, computed before rounding; positive values indicate lower recall after deletion. Full values vary with the paired evaluation subset.

Removed field	Full	Ablated	Δ (pp)
Affected entity	64.3	57.1	+7.1
Trigger	64.3	71.4	−7.1
Failure mechanism	65.5	55.2	+10.3
Potential impact	65.5	65.5	0.0
Evidence needed	64.3	58.9	+5.4
Verification plan	65.5	50.0	+15.5
Risk hypothesis	64.3	62.5	+1.8

Removing the affected entity and evidence needed yields smaller drops of 7.1 and 5.4 points. Removing the risk-hypothesis summary changes recall by 1.8 points, while removing potential impact leaves it unchanged. Removing the trigger increases recall by 7.1 points. Related information may remain in other fields, so these results describe each field's additional contribution within the full representation, rather than a definitive ranking of field importance.

6 Discussion

The results suggest that risk coverage and investigation effectiveness capture different aspects of code review. A broader candidate pool creates more opportunities to identify issues, but those opportunities must receive attention and lead to sufficient evidence. This distinction helps interpret the different outcomes of RQ2 and RQ3: selecting relevant hypotheses from an existing pool improves recall, whereas generating and investigating hypotheses within a shared budget does not show the same benefit. The field ablations further suggest that explaining a failure mechanism and specifying how to investigate it can make guidance useful. Together, these observations motivate assessing hypotheses through both their relevance before investigation and the findings obtained afterward.

7 Limitations

The benchmark draws on historical concerns from agent-authored PRs, and finding-level evaluation focuses on risks with accessible verification evidence. Manual review strengthens reference quality, but the reference set may remain incomplete and automated finding judgments may contain errors. Recall therefore measures coverage of the curated reference concerns. The results apply to the tested models, resource budgets, and evaluation cohorts, including the jointly evaluable subsets used for paired comparisons. Oracle selection relies on Gold labels, and Gold Guidance uses a separate retrospective evaluation; these comparisons characterize the potential of informed guidance within the tested settings.

8 Conclusion

We introduced the Risk Hypothesis Benchmark, comprising 300 structured reference risks grounded in review-time repository evidence. The benchmark connects risk formulation and selection to downstream code-review findings. Our experiments show broader observed risk coverage with explicit hypothesis generation, model-dependent ranking gains, and higher finding recall when relevant candidates receive priority. These benefits do not automatically translate into better end-to-end review under a fixed total budget. The guidance comparisons and field ablations further suggest that the information supplied for investigation matters. By making intermediate concerns explicit and testable, the benchmark provides concrete evaluation targets for understanding and improving how LLM reviewers turn potential risks into supported findings.

Reproducibility Statement

The benchmark data, construction prompts and scripts, and experimental configurations and analysis files are available in the anonymous supplementary artifact at `https://anonymous.4open.science/r/risk-hypothesis-iclr2027/`. Appendix A describes the materials, and Appendix B gives the evaluation conventions and resource limits.

AI Use Statement

We used generative AI tools for language editing and translation, assistance with code implementation and data formatting, and checking and interpreting experimental results. Model-assisted benchmark construction and automated evaluation are described in the main paper. The authors take responsibility for the final text, code, data, and reported findings.

References

Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. AIDev: Studying AI coding agents on GitHub. *arXiv preprint arXiv:2602.09185*, 2026. URL https://arxiv.org/abs/2602.09185.

Chunhua Liu, Hong Yi Lin, and Patanamon Thongtanunam. Too noisy to learn: Enhancing data quality for code review comment generation. *arXiv preprint arXiv:2502.02757*, 2025. URL https://arxiv.org/abs/2502.02757.

Junyi Lu, Xiaojia Li, Zihan Hua, Lei Yu, Shiqi Cheng, Li Yang, Fengjun Zhang, and Chun Zuo. DeepCRCEval: Revisiting the evaluation of code review comment generation. *arXiv preprint arXiv:2412.18291*, 2024. URL https://arxiv.org/abs/2412.18291.

Atharva Naik, Marcus Alenius, Daniel Fried, and Carolyn Rose. CRScore: Grounding automated evaluation of code review comments in code claims and smells. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9049–9076. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.457. URL https://aclanthology.org/2025.naacl-long.457/.

Kristen Pereira, Neelabh Sinha, Rajat Ghosh, and Debojyoti Dutta. CR-Bench: Evaluating the real-world utility of AI code review agents. *arXiv preprint arXiv:2603.11078*, 2026. URL https://arxiv.org/abs/2603.11078.

Kla Tantithamthavorn, Hong Yi Lin, Patanamon Thongtanunam, Wachiraphan Charoenwet, Minwoo Jeong, and Ming Wu. HalluJudge: A reference-free hallucination detection for context misalignment in code review automation. *arXiv preprint arXiv:2601.19072*, 2026. URL https://arxiv.org/abs/2601.19072.

Ziliang Wang, Ge Li, Jia Li, Hao Zhu, and Zhi Jin. VulAgent: Hypothesis-validation driven multi-agent architecture for vulnerability detection. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 18598–18616. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.928. URL https://aclanthology.org/2026.findings-acl.928/.

Daoan Zhang, Shuo Zhang, Zijian Jin, Jiebo Luo, Shengyu Fu, and Elsie Nallipogu. Sphinx: Benchmarking and modeling for LLM-driven pull request review. *arXiv preprint arXiv:2601.04252*, 2026a. URL https://arxiv.org/abs/2601.04252.

Lei Zhang, Yongda Yu, Minghui Yu, Xinxin Guo, Zhengqi Zhuang, Guoping Rong, Dong Shao, Haifeng Shen, Hongyu Kuang, Zhengfeng Li, Boge Wang, Guoan Zhang, Bangyu Xiang, and Xiaobin Xu. AACR-Bench: Evaluating automatic code review with holistic repository-level context. *arXiv preprint arXiv:2601.19494*, 2026b. URL https://arxiv.org/abs/2601.19494.

A Benchmark and Supplementary Materials

The benchmark contains 300 Gold risks from 223 review cases. The anonymous artifact provides the risk records, review-case manifest, construction prompts and scripts, and experimental configurations and analysis files:

`https://anonymous.4open.science/r/risk-hypothesis-iclr2027/`

Each risk contains the seven semantic fields and provenance identifiers. The review-case manifest identifies the public repository, pull request, and reviewed revision. Historical comments and subsequent changes are used during reference construction but are withheld from the evaluated reviewer.

The construction materials specify the evidence boundary, admission criteria, and treatment of unresolved premises. A hypothesis must identify an affected entity and a plausible failure mechanism grounded in review-time evidence. An unresolved premise must be stated as a condition and paired with a check that can support or refute it. Evidence-backed concerns remain hypotheses until verification supports a finding.

The package includes an offline manifest check, `python3 scripts/validate_artifact.py`. Live experiments also require the public source revisions, the runner dependencies, and access to the specified model providers. Provider credentials are supplied through environment variables. The included scripts record the experimental procedures; exact outputs can vary with model sampling and provider updates.

B Evaluation Details

B.1 Metrics and comparison sets

PR-macro Strict Finding Recall averages recall across PRs, while Gold-micro recall divides the total number of distinct matched reference risks by the total number of eligible reference risks. A reference contributes at most one hit per PR and condition. Validity precision is the proportion of judged findings judged valid; a valid finding need not match a curated reference risk. PR hit rate and valid-PR rate measure the proportions of PRs with at least one strict Gold match and at least one valid finding, respectively.

RQ1, RQ2, and the primary RQ4 contrasts use results evaluable in both conditions of each comparison. Legal empty outputs contribute zero recall; unavailable outputs or judgments are excluded from these paired content comparisons. Consequently, the reported differences apply to the corresponding evaluable subsets. RQ3 instead reports workflow recall and PR-level rates over its fixed cohort of 60 PRs and 68 eligible reference risks. A workflow that delivers no admissible finding contributes no hit, without treating its execution status as a judgment that the code contains no relevant risk.

The primary RQ1 paired sets contain 156 cases and 218 Gold risks for Flash, 158 and 223 for Pro, 37 and 53 for GLM, and 45 and 61 for Qwen Max. Each model is compared within its own paired set. For RQ2, the Oracle comparisons use 53 paired PRs with GPT-5.6-sol and 30 with Pro. Pro's Random, BM25, and Generic LLM comparisons use 25, 25, and 20 paired PRs, respectively. Selection coverage is computed separately on cases with complete candidate-to-Gold labels. These sets should not be combined into a single ranking of models or selectors.

B.2 Model roles and evaluation

RQ2 fixes the Flash-generated candidate pool and varies selection within each verifier setting. The selected candidates retain their content and are presented in canonical order without strategy labels. RQ3 uses the same backbone for Self's planning and verification stages. RQ4 supplies reference-derived guidance to Flash and deletes one field at a time, leaving the remaining fields unchanged.

The DeepSeek RQ1 comparisons and the controlled finding evaluations use GPT-5.6-sol with high reasoning for automated judgments. Gold Guidance is evaluated separately, post hoc and without blinding. The Qwen extension also combines existing evaluator judgments with post-hoc judgments from an interactive coding assistant session. These supplementary comparisons do not establish an

independent replication under a common evaluator. Automated judgments may differ from expert assessment.

B.3 RQ3 RESOURCE LIMITS

Table 6 gives the shared resource limits. Direct receives the full review budget. Self divides it between planning and verification, with no transfer of unused planning allocation. Gold Guidance receives only the verification allocation and incurs no hypothesis-generation stage.

Table 6: RQ3 resource limits per PR. Token limits are in thousands; wall time is in seconds. Charged tokens follow the runner's accounting rule and are not monetary cost estimates.

Resource	Direct	Self plan	Self verify	Gold Guidance
Charged tokens	100	30	70	70
Generated tokens	30	9	21	21
Tool calls	20	6	14	14
Tool-return tokens	30	9	21	21
Wall time	1,200	360	840	840

The main Self and Gold Guidance results require evidence to be admissible under the original review-trajectory rule. A separate snapshot-content analysis checks findings against the frozen code snapshot without the same trajectory restriction. Under this alternative, Gold Guidance's PR-macro recall is 39.17% for Flash and 43.33% for Pro, compared with 34.17% and 41.67% in the main analysis. These values answer different evaluation questions and are reported separately.

C ADDITIONAL RQ1 RESULTS

Table 7 reports case-macro ranking metrics for the DeepSeek comparisons. Flash improves at each listed cutoff. Pro's MRR changes little, and its lower NDCG at ranks one and five contrasts with higher NDCG at rank ten. The broader coverage therefore does not imply consistently better ordering near the top of the list.

Table 7: Case-macro Strict ranking metrics. Each model uses the paired population of its primary DH–Review-Norm comparison. MRR and NDCG are reported on a 0–1 scale.

Model	Condition	MRR	NDCG@1	NDCG@5	NDCG@10
Flash	Review-Norm	.3507	.2500	.3696	.3948
	DH	.3867	.2756	.3826	.4466
Pro	Review-Norm	.2868	.2215	.3025	.3153
	DH	.2887	.1835	.2893	.3457

Table 8 gives the control against raw Direct Review. Flash, Pro, and Qwen have higher observed Strict Recall@10 under DH; GLM has lower recall in this comparison. Each row uses the joint evaluation set for DH and raw DR, which can differ from the DH–Review-Norm set.

Table 8: Gold-micro Strict Recall@10 (%) against raw Direct Review. Δ is DH minus DR, calculated before rounding.

Model	Cases / Gold	DR	DH	Δ (pp)
Flash	157 / 219	51.60	59.36	+7.76
Pro	161 / 226	38.94	47.79	+8.85
GLM	29 / 40	47.5	40.0	-7.5
Qwen Max	46 / 62	46.8	54.8	+8.1

Preserving candidate count and order during normalization does not establish semantic equivalence. On Pro's common raw/normalized cases, normalization changes Strict Recall@10 by -3.54 points.

In GLM's three-condition common set, Strict Recall@10 is 45.2% for DR, 35.7% for Review-Norm, and 40.5% for DH. The GLM conclusion thus remains sensitive to the baseline even when the evaluation set is held fixed. These comparisons concern reference coverage, not proof of defects or correctness of every risk field.

D Additional RQ4 Results

The seven-field ablation uses 30 development or previously exposed PRs with 35 eligible Gold risks. Each Full–ablation comparison uses 28 or 29 jointly evaluable PRs. Table 9 additionally holds the evaluation set fixed across all eight conditions, using the 25 PRs with results in every condition.

Table 9: RQ4 results (%) on the common set across all eight conditions. Macro R and Micro R denote PR-macro and Gold-micro Strict Finding Recall. Precision is pooled finding validity precision on this same set.

Guidance	Macro R	Micro R	Precision
Full	60.0	55.2	43.5
Without affected entity	52.0	48.3	49.0
Without trigger	68.0	62.1	50.0
Without failure mechanism	48.0	44.8	51.9
Without potential impact	64.0	58.6	57.7
Without evidence needed	58.0	51.7	45.0
Without verification plan	50.0	48.3	44.2
Without risk hypothesis	58.0	51.7	47.7

The common-set analysis also shows lower recall after removing the failure mechanism or verification plan. However, the size and ordering of the decreases change: the mechanism deletion has the larger drop on this set. Precision does not follow the same pattern as recall. For example, removing the mechanism reduces macro recall from 60.0% to 48.0% while increasing precision from 43.5% to 51.9%.

Uncertainty is substantial. For the pairwise comparisons in the main text, the 95% interval for Full minus plan deletion is $[-4.0, 34.6]$ percentage points, and that for Full minus mechanism deletion is $[0.0, 23.1]$. These intervals use 10,000 repository-cluster bootstrap samples, retaining PRs from the same repository together. They are pointwise intervals, without adjustment for the seven comparisons. The results describe additional contributions within this representation and cohort; they do not establish a universal ranking or show that a field with no observed loss is unnecessary.