TRUST BEFORE AUTONOMY: PRE-OUTCOME SELECTIVE ROUTING FOR CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents act before their final correctness is known. We study *pre-outcome selective autonomy*: whether an exact software change should proceed automatically, be reviewed, or be withheld using only evidence available before grading. OPENCODER X binds the candidate to uncertainty, evidence provenance, verification status, and run identity while enforcing the boundary $y \notin z$ between pre-outcome information and the eventual benchmark label. A frozen policy then maps that record to ACCEPT, REVIEW, or ABSTAIN. Across 128 labeled SWE-bench Verified candidates, two failure modes emerge. In a matched 68-candidate development pilot, Guard changes resolved-task rate from 12/17 to 13/17 for the Codex client but from 13/17 to 11/17 for the Claude Code client; paired inference is inconclusive, and the effects differ in sign. On a disjoint 60-candidate calibration split, both clients resolve 20/30 tasks, yet every candidate receives raw failure risk 0.35 (AUROC 0.50; Brier 0.2225). Isotonic calibration maps the single support point to failure probability $1/3$, after which the frozen policy abstains on every candidate. The failure is therefore not merely miscalibration: the score contains no instance-level ordering to exploit. More generally, deterministic scalar calibration cannot separate candidates that are tied by the underlying score; K distinct score values yield at most $K + 1$ threshold-defined empirical coverage regimes. The contribution is a leakage-resistant protocol for coding-agent authorization together with an empirical diagnosis of when selective routing fails because the pre-outcome representation itself lacks discrimination.

1 INTRODUCTION

Coding agents now navigate repositories, edit multiple files, invoke tools, and iterate on execution feedback. Benchmarks such as SWE-bench make repository-level task success measurable (Jimenez et al., 2024), while recent systems and datasets study agent interfaces, repair, repository context, tests, professional software work, and scalable training/verification (Yang et al., 2024; Zhang et al., 2024; Hai et al., 2025; Mündler et al., 2024; Miserendino et al., 2025; Pan et al., 2025; Yang et al., 2025). These advances primarily answer a capability question: *can the agent complete the task?*

Deployment requires a different decision before that outcome exists: *does the evidence available for this particular attempt justify autonomy?* A plausible patch may violate a repository invariant, rely on stale evidence, or pass only incomplete checks; conversely, a correct patch may arrive with weak or missing confidence signals. The object of interest is therefore not model confidence in isolation, but a pre-outcome authorization decision tied to the exact software change being authorized.

This problem connects coding-agent evaluation to selective prediction, calibration, learning to defer, and uncertainty quantification (Geifman & El-Yaniv, 2017; 2019; Guo et al., 2017; Mozannar & Sontag, 2020; Ovadia et al., 2019). Agent settings add three complications. First, telemetry is heterogeneous and often missing, and uncertainty can evolve over an interaction trajectory (Oh et al., 2026). Second, verification evidence may be self-reported by the same system under evaluation. Third, the candidate itself can change after an assessment, so a risk score can silently become detached from the action it purportedly authorized. Benchmark evaluation introduces a fourth hazard: if correctness is joined before the trust state is frozen, the decision can leak the answer it is supposed to predict.

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 study these issues through OPENCODER X, a trust layer separated from generation. Its central invariant is $y \notin z$: final correctness y never enters pre-decision telemetry z. An exact candidate c is assessed from pre-outcome evidence, then risk and routing are frozen before benchmark evaluation. This boundary separates three estimands that are often conflated: *prediction* (does risk rank failures?), *selection* (does routing reduce failure at useful coverage?), and *intervention* (does exposing a guard during generation change the candidate or its correctness?).

Our contributions are:

- **Pre-outcome selective autonomy as a candidate-level decision problem.** We bind uncertainty, provenance, verification, missingness, and run identity to the exact software change being authorized while excluding outcome labels.
- **A leakage-resistant protocol that separates prediction, selection, and intervention.** Candidate identity and trust state are frozen before correctness, and retained traces verify whether an intended Guard treatment actually reached the evaluated candidate.
- **A completed empirical diagnosis across 128 labeled candidates.** Guard effects are heterogeneous across client conditions, while the disjoint calibration split exhibits complete score-support collapse. The latter exposes a basic operational limit: calibration can remap a score but cannot manufacture ordering that the representation never encoded.

We ask three questions: whether pre-outcome risk discriminates failures; whether live Guard treatment changes correctness consistently across client conditions; and whether candidate, treatment, assessment, and later outcome can be kept auditable without leaking correctness into the decision.

2 PRE-OUTCOME SELECTIVE AUTONOMY

2.1 PROBLEM FORMULATION

Let a coding system receive task x and produce candidate software change c. Before the final benchmark oracle executes, the environment may expose telemetry z: repeated generations, token-level uncertainty when available, retrieved evidence, provenance metadata, compilation/test status, static checks, or trace-level signals. Final correctness $y \in \{0, 1\}$ is unavailable to the trust layer at this stage.

We seek a pre-outcome risk function

$$r = f_\theta(c, z) \in [0, 1], \qquad y \notin z, \tag{1}$$

and a routing policy

$$d = \pi(r, z) \in \{\textsc{Accept}, \textsc{Review}, \textsc{Abstain}\}. \tag{2}$$

Equations 1–2 encode the central boundary: only after (c, z, r, d) are frozen may the external benchmark evaluator produce y and join it to the research record. For a calibrated failure probability $\tilde{r}$, the three-way policy uses frozen thresholds $\tau_A < \tau_R$:

$$d(\tilde{r}) = \begin{cases} \textsc{Accept}, & \tilde{r} \le \tau_A, \\ \textsc{Review}, & \tau_A < \tilde{r} \le \tau_R, \\ \textsc{Abstain}, & \tilde{r} > \tau_R. \end{cases} \tag{3}$$

Equation 3 makes the operational action explicit, while the label boundary keeps failure prediction, selective routing, and intervention effects as distinct estimands rather than different names for the same outcome.

For a calibrated failure probability $\tilde{r}_i$ and threshold τ, autonomous coverage and selective risk are

$$C(\tau) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[\tilde{r}_i \le \tau], \tag{4}$$

$$R(\tau) = \frac{\sum_i (1 - y_i) \mathbf{1}[\tilde{r}_i \le \tau]}{\sum_i \mathbf{1}[\tilde{r}_i \le \tau]}, \quad \text{when } \sum_i \mathbf{1}[\tilde{r}_i \le \tau] > 0. \tag{5}$$

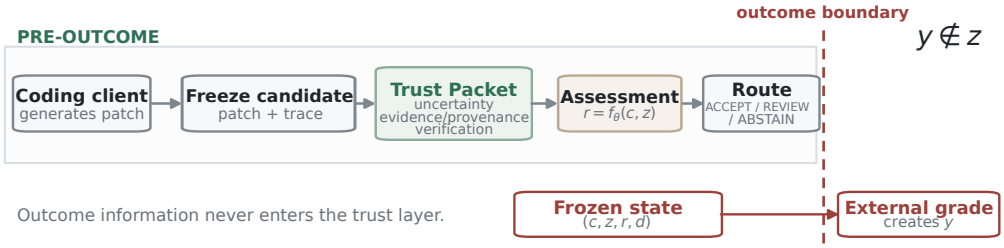

Figure 1: **Pre-decision trust pipeline.** Generation remains upstream. OPENCODER X assesses only the exact candidate and pre-outcome information; benchmark correctness joins only after the assessment is frozen.

Equation 5 makes clear that useful selective autonomy requires more than calibrated probabilities: the score must create an ordering that allows $R(\tau)$ to decrease at nontrivial $C(\tau)$. At zero coverage, selective risk is left undefined rather than assigned an artificial value. We therefore track the empirical *score support*

$$K(r_{1:n}) = |\{r_1, \ldots, r_n\}|, \tag{6}$$

which measures how many distinct risk values are available to a deterministic threshold policy. Calibration quality and score support answer different questions: the former concerns probability level, while the latter limits how finely candidates can be separated by risk.

2.2 TRUST PACKET AND CANDIDATE BINDING

Figure 1 summarizes the system boundary. Generation remains upstream. A thin adapter retains the exact candidate and native trace, constructs a provider-neutral Trust Packet, and binds the assessment to that candidate. Missing signals remain missing. Final rewards, grader verdicts, hidden-test outcomes, and aliases of those outcomes are rejected from pre-decision inputs.

The packet separates three evidence families. **Uncertainty** may include token-distribution entropy when exposed or repeated-sample disagreement when it is not; these are treated as imperfect measurements rather than confidence ground truth, consistent with semantic and consistency-based uncertainty work (Kuhn et al., 2023; Farquhar et al., 2024; Manakul et al., 2023). **Evidence/provenance** records source identity, relevance/quality metadata, revisions, and integrity digests when available. **Verification** records compilation, tests, type checking, static analysis, or other checks, but distinguishes independently collected evidence from caller-provided self-attestation. This distinction matters because LLM-based verification can itself be unstable and should not be treated as an oracle (Jiang et al., 2026).

The implemented baseline is deliberately transparent rather than claimed as a learned optimal risk estimator. It combines normalized uncertainty risk u^*, evidence risk e^*, and verification risk v^* using frozen weights

$$r_{\text{base}} = 0.45u^* + 0.25e^* + 0.30v^*, \tag{7}$$

with conservative fixed fallbacks when a signal family is unavailable. Equation 7 is fixed before outcome inspection; it is a transparent engineering baseline, not a learned posterior. The point of the completed experiments is therefore diagnostic: do these pre-outcome signals actually discriminate failures, and does calibration create useful routing behavior?

2.3 PREDICTION, SELECTION, AND INTERVENTION ARE DIFFERENT

A risk score can predict errors without changing generation; an intervention can change generation without improving risk estimation. We therefore distinguish: **prediction**, whether risk ranks incorrect attempts above correct ones; **selection**, whether routing high-risk attempts reduces residual failure at useful coverage; and **intervention**, whether exposing the trust layer during generation changes task success. The development Guard study probes intervention, while the disjoint calibration study probes whether the frozen score and its calibrated image contain enough support for meaningful selection.

Table 1: Completed experimental roles. Outcome labels are joined only after the candidate and pre-outcome trust state are frozen.

Study	N	Scientific role
Matched development pilot	68	intervention audit; failure-ranking diagnostic
Disjoint calibration split	60	calibration; score-support and routing diagnostic

3 Experimental Protocol

3.1 Clients, benchmark, and provenance

Completed labeled experiments use SWE-bench Verified with two coding-client conditions: a Codex CLI condition and a Claude Code condition. The retained runs use native client executions under a frozen routing/provenance protocol, so results are attributed to the evaluated client conditions rather than to direct first-party model behavior. Client version, route/provenance, task, repository revision, tool permissions, and resource constraints are retained as condition metadata. SWE-bench supplies the external repository-level correctness signal (Jimenez et al., 2024); we use it as a controlled environment for trust analysis rather than as a universal measure of software-engineering ability.

Every counted run must yield an exact candidate and a native trace. The assessment record is bound to that candidate and frozen before correctness is joined. Runs that terminate before satisfying the predefined candidate contract are retained as incidents rather than silently converted into benchmark failures. Likewise, configuring Guard does not establish that Guard was delivered: intervention analysis requires trace evidence that the intended treatment occurred and that the assessed candidate corresponds to the final software change.

3.2 Completed studies

The development pilot uses 17 tasks under four matched cells: Codex client Observation, Codex client Guard, Claude Code client Observation, and Claude Code client Guard (68 candidates). Observation leaves generation unchanged and assesses the frozen attempt afterward. Guard places OpenCoder X in the action path and may affect the workflow. This study probes intervention behavior and protocol fidelity; its sample size does not support a broad causal claim.

A disjoint calibration split contains 30 tasks per client (60 candidates). It is used to diagnose the frozen risk score, fit per-client isotonic calibration, and apply the preregistered routing thresholds. The development study also motivates an explicit treatment-validity rule: configuration alone is insufficient evidence that Guard affected the evaluated candidate. A run enters matched treatment analysis only when the retained native trace confirms treatment delivery and the assessed candidate corresponds to the final software change.

Together the two completed studies contribute 128 labeled candidates. Their roles are deliberately different: the matched pilot tests intervention behavior, whereas the calibration split tests discrimination, probability calibration, and whether the score has enough empirical support to induce nontrivial routing.

3.3 Metrics, calibration, and baselines

Failure is the positive class. We report AUROC for discrimination, score support K from Eq. 6, and Brier score (Brier, 1950) when risk is interpreted probabilistically. AUPRC and AURC remain computable under ties, but when $K = 1$ they are dominated by prevalence and tie conventions rather than usable ranking; we therefore do not interpret them as evidence of selectivity on the calibration split. The calibration split fits per-client isotonic (PAV) calibration because it preserves score ordering without imposing a parametric link. The calibration map and routing thresholds are fixed within the calibration study before any downstream interpretation of correctness-dependent metrics.

For selective autonomy we report coverage and the routing induced by the frozen thresholds. This complements calibration because an all-abstain policy can appear conservative while providing no

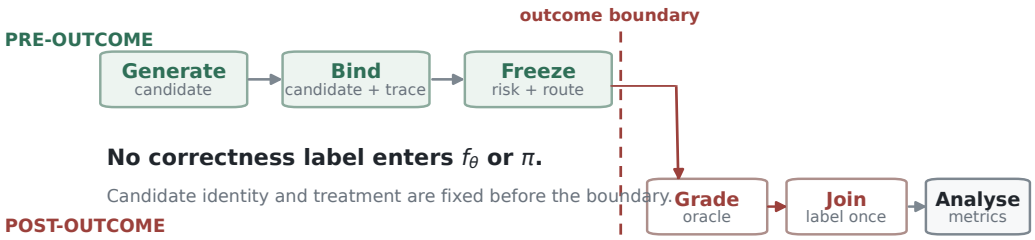

Figure 2: **Leakage-resistant evaluation lifecycle.** Candidate generation, candidate binding, and the trust decision occur entirely before the outcome boundary. External grading creates y only after the pre-outcome state is frozen; the one-time label join is used for analysis, never as an input to assessment or routing.

Table 2: Matched Observation→Guard effects on 17 development tasks per client. CI is the paired-bootstrap 95% interval for the accuracy difference.

Client	Obs.	Guard	Δ pp	I/R/T	McNemar p	95% CI
Codex	12/17	13/17	+5.9	2/1/14	1.00	$[-11.8, 23.5]$
Claude Code	13/17	11/17	-11.8	0/2/15	0.50	$[-29.4, 0.0]$

autonomous utility. Risk-control methods that target adaptive guarantees are relevant comparators (Blot et al., 2025), but no distribution-free safety guarantee is claimed here.

4 Empirical Results: Intervention Heterogeneity and Support Collapse

4.1 Guard effects are client-dependent, not uniformly beneficial

All 68 pilot candidates and their pre-label trust states were frozen before official grading; each candidate was graded once and trust was not recomputed afterward. Table 2 reports paired outcomes.

Figure 3 visualizes the paired changes. Codex Guard improves two tasks, regresses one, and ties on fourteen. Claude Code Guard improves none, regresses two, and ties on fifteen. The paired tests are inconclusive and the intervals overlap zero. Thus the pilot does *not* support a client-independent Guard correctness benefit.

The risk score yields a sharper negative result. Both Observation cells and Claude Code Guard assign one tied risk value across all 17 tasks and therefore have error-AUROC 0.50. Codex Guard changes routing but yields error-AUROC 0.346, below 0.5 under the frozen score orientation. The pilot demonstrates the assessment-before-outcome workflow and exposes heterogeneous intervention behavior, but it does not validate the current risk estimator.

4.2 Calibration cannot recover selectivity from collapsed support

The clearest failure mode appears on the disjoint calibration split. Each client resolves 20/30 tasks (66.7%), yet every candidate receives the same raw risk, 0.35. Thus $K = 1$ for both clients—one support point across 30 candidates—and failure AUROC is exactly 0.50. Table 3 summarizes the frozen diagnostics.

PAV calibration fits one block per client at the empirical failure probability $1/3$. Under the frozen thresholds (ACCEPT if $p_{\text{fail}} \leq 0.10$, REVIEW if $p_{\text{fail}} \leq 0.30$, otherwise ABSTAIN), every calibration candidate abstains. Figure 4 makes this distinction explicit.

Support consequence. Let the finite sample contain K distinct raw scores and let g be any deterministic scalar calibration map. The calibrated sample can contain at most K distinct values because each raw value has only one image under g. A scalar threshold can therefore change the

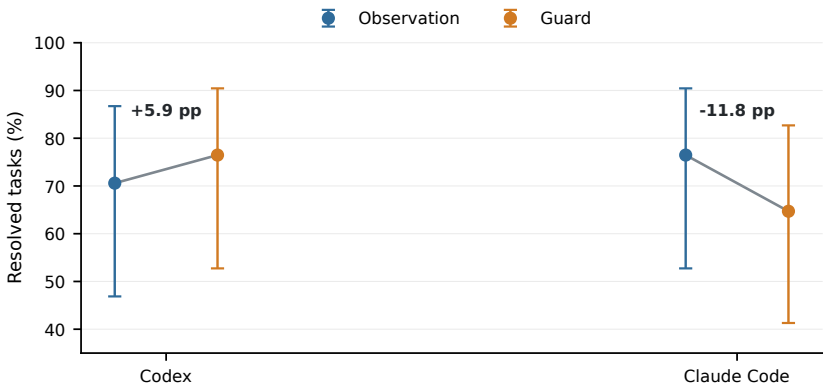

Figure 3: **Matched development-pilot outcomes.** Observation and Guard are compared on the same 17 tasks within each client. Wilson intervals are descriptive; paired inference uses the task-level transitions in Table 2.

Table 3: Calibration-split diagnostics. "A/R/Ab" denotes ACCEPT/REVIEW/ABSTAIN counts under the frozen calibrated policy.

Client	Resolved	Raw support	AUROC	Brier	PAV p_{fail}	A/R/Ab
Codex	20/30	$\{0.35\}$	0.500	0.2225	0.333	0/0/30
Claude Code	20/30	$\{0.35\}$	0.500	0.2225	0.333	0/0/30

accepted set only when it crosses one of those values, yielding at most $K + 1$ empirical coverage regimes including the empty and full sets. Candidates tied under the raw score remain tied after calibration. In particular, when $K = 1$ and both outcome classes occur, tie-aware AUROC is 0.5 and binary autonomous coverage can only be 0 or 1.

This consequence is elementary, but it is the key diagnostic here. Calibration can correct probability level; it cannot create missing discrimination. Because the calibration split has $K = 1$, isotonic calibration moves the common score to the empirical failure frequency while leaving only the two trivial binary coverage regimes. The operative failure is therefore in the pre-outcome representation, not in the choice of scalar calibrator.

5 What Reliable Selective Autonomy Must Learn

The completed studies point to a failure upstream of calibration: the current pre-outcome representation often lacks attempt-specific variation. In three of four pilot cells there is no within-cell risk variation, and the calibration split collapses exactly to risk 0.35 for both clients. This suggests three design principles for future learned autonomy models.

Treat missingness as information. Provider-neutral interfaces inevitably observe different telemetry across coding systems. Fixed fallback values are transparent, but repeated fallback patterns can collapse a score. Missingness indicators, provenance quality, and evidence-acquisition decisions should therefore be modeled explicitly rather than silently compressed into the same constant.

Distinguish verification from self-attestation. A caller reporting that a test passed is not equivalent to an independently executed check bound to the exact candidate. Verifier outputs are also noisy measurements: recent code-judge evaluations show substantial instability under seemingly innocuous perturbations (Jiang et al., 2026). Verification evidence should therefore carry provenance and empirically estimated reliability rather than being treated as an oracle.

Exploit trajectory-level evidence without crossing the label boundary. Agent uncertainty need not be a property of the final patch alone. Interactive agents revise plans, gather repository evidence, invoke tools, and encounter failures over time. Recent agent-UQ work emphasizes heterogeneous uncertainty targets and temporal dynamics (Oh et al., 2026). Learned risk models can use these

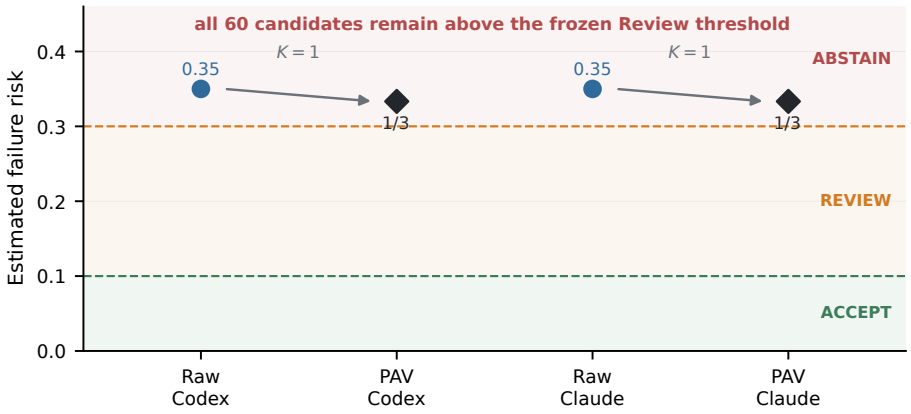

Figure 4: **Calibration-split score-collapse diagnostic.** Raw risk is 0.35 for every candidate. Isotonic calibration maps the single support point to $p_{\text{fail}} = 1/3$, above the frozen Review threshold, producing zero autonomous coverage.

pre-outcome trajectory features, but feature design and model selection must remain separated from the correctness labels used for final evaluation.

A selective-autonomy pipeline should therefore diagnose *support* before interpreting calibration as evidence of useful routing. If the score does not distinguish attempts, increasingly sophisticated calibrators cannot recover a useful risk–coverage frontier; the remedy must change the evidence or representation, not only the probability mapping.

6 Related Work

Software-engineering agents and benchmarks. SWE-bench (Jimenez et al., 2024) catalyzed repository-level evaluation, followed by work on agent interfaces (Yang et al., 2024), autonomous repair (Zhang et al., 2024), agent behavior (Xia et al., 2025), repository context (Hai et al., 2025), test-focused evaluation (Mündler et al., 2024), professional software tasks (Miserendino et al., 2025), refreshed code evaluation (Jain et al., 2025), and scalable training/verification data (Pan et al., 2025; Yang et al., 2025). These works primarily evaluate capability after a task outcome is produced. We study the complementary pre-outcome question: whether the evidence available for a specific attempt is sufficient to justify autonomy before that outcome exists.

Uncertainty, abstention, and risk control. Selective classification and SelectiveNet formalize prediction with a reject option (Geifman & El-Yaniv, 2017; 2019); learning to defer extends the setting to expert routing (Mozannar & Sontag, 2020). Calibration work shows that neural confidence can be miscalibrated (Guo et al., 2017), especially under shift (Ovadia et al., 2019). Semantic entropy and black-box consistency provide alternatives to raw token confidence for language models (Kuhn et al., 2023; Farquhar et al., 2024; Manakul et al., 2023). More recent work broadens UQ to interactive agents (Oh et al., 2026) and develops adaptive risk-control methods (Blot et al., 2025). Our contribution is to expose how these ideas interact with candidate identity, evidence provenance, verifier reliability, and treatment fidelity in repository-level coding agents.

Tool-using agents and verification. ToolEmu shows that LM-agent risk depends on interaction with external tools and environments, not only response quality (Ruan et al., 2024). CodeJudgeBench shows that LLM-based code judges are themselves sensitive to perturbations (Jiang et al., 2026). OpenCoder X operationalizes a complementary control point: a provider-neutral pre-outcome boundary that records why a candidate was accepted, reviewed, or withheld and preserves the evidence required to replay that decision.

7 Limitations and Scope

The present evidence does not establish production calibration, universal failure prediction, formal safety, or a general correctness advantage from Guard. The matched intervention pilot is small, the first risk function is deliberately transparent rather than learned, and isotonic calibration is evaluated on the same disjoint split used to fit it. The paper should therefore be read as a methodological and diagnostic study of pre-outcome autonomy, not as evidence that the current score is deployment-ready.

The labeled experiments are limited to two coding-client conditions and SWE-bench Verified. Provider-neutrality is an interface property, not proof of transfer across models, repositories, languages, benchmarks, or time. A routing layer can reduce or reallocate autonomy, but it is not a sandbox and does not make generated code safe to execute. High-stakes deployment would require stronger independent verification, environment isolation, access control, monitoring, and governance.

These limitations do not alter the central observation: useful selective autonomy requires discriminative pre-outcome evidence. When empirical score support collapses, post-hoc calibration can improve probability matching while leaving the autonomy decision operationally non-selective.

8 Conclusion

We introduced pre-outcome selective autonomy for coding agents and instantiated it in OpenCoder X, a provider-neutral trust layer that separates generation, candidate-bound assessment, routing, and later correctness. Across 128 labeled candidates, the two completed studies expose different weaknesses. Guard changes correctness in opposite directions across the two client conditions, with inconclusive paired inference. More importantly, the disjoint calibration split collapses to one risk value for every candidate. Isotonic calibration can match the observed failure frequency, but only by preserving that tie and producing an all-abstain policy.

The lesson is simple. Calibration is not a substitute for representation. If pre-outcome evidence does not order attempts by risk, a scalar probability map cannot recover selective autonomy after the fact. Reliable authorization therefore requires joint evaluation of discrimination, probability calibration, score support, treatment fidelity, and operational coverage under a strict assessment-before-outcome protocol.

Reproducibility Statement

The main text specifies the label boundary, candidate-binding rule, client provenance, development and calibration roles, frozen routing thresholds, metrics, and treatment definitions. Exact candidates, native traces, assessment records, and run manifests are retained so the pre-outcome decision can be replayed deterministically and checked against the later correctness join. Identifying repository links are omitted from the double-blind manuscript; an anonymized artifact bundle can accompany review.

Ethics Statement

This work studies control and evaluation of coding-agent autonomy and does not involve human subjects or sensitive personal data. The main foreseeable risk is misuse of a trust score as a safety guarantee. We therefore distinguish predictive risk, workflow routing, independent verification, and formal assurance, and limit claims to the evaluated clients, benchmark, versions, and frozen conditions.

References

Vincent Blot, Anastasios Nikolas Angelopoulos, Michael Jordan, and Nicolas J-B. Brunel. Automatically adaptive conformal risk control. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 19–27. PMLR, 2025.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950. doi: 10.1175/1520-0493(1950)078⟨0001:VOFEIT⟩2.0.CO;2.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625–630, 2024. doi: 10.1038/s41586-024-07421-0.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2151–2159, 2019.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, 2017.

Nam Le Hai, Dung Manh Nguyen, and Nghi D. Q. Bui. On the impacts of contexts on repository-level code generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1496–1524, 2025. doi: 10.18653/v1/2025.findings-naacl.82.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

Hongchao Jiang, Yiming Chen, Yushi Cao, Hung-yi Lee, and Robby T. Tan. CodeJudgeBench: Benchmarking LLM-as-a-judge for coding tasks. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19416–19448. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.888.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, 2023. doi: 10.18653/v1/2023.emnlp-main.557.

Samuel Miserendino, Michele Wang, Tejal Patwardhan, and Johannes Heidecke. SWE-lancer: Can frontier LLMs earn $1 million from real-world freelance software engineering? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 44412–44450, 2025.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7076–7087, 2020.

Niels Mündler, Mark Niklas Müller, Jingxuan He, and Martin Vechev. SWT-bench: Testing and validating real-world bug-fixes with code agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 81857–81887, 2024. doi: 10.52202/079017-2601.

Changdae Oh, Seongheon Park, To Eun Kim, Jiatong Li, Wendi Li, Samuel Yeh, Sean Du, Hamed Hassani, Paul Bogdan, Dawn Song, and Sharon Li. Uncertainty quantification in LLM agents: Foundations, emerging challenges, and opportunities. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16219–16250. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.738.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, pp. 13991–14002, 2019.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-gym. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737, 2025.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, 2024.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025. doi: 10.1145/3715754.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024. doi: 10.52202/079017-1601.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 107402–107452, 2025. doi: 10.52202/085713-3239.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. AutoCodeRover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1592–1604, 2024. doi: 10.1145/3650212.3680384.