

PECA v3: Evidence-Gated Anticipatory Execution: Separating Requirement Compliance, Predictive Utility, and Commit Authorization in AI Agents

RM369

Abstract

Anticipation can help an AI agent prepare useful work before a requirement is fully specified, but a predictor can also create the very obligations the agent then tries to satisfy. PECA v3 refactors Prefix-Conditioned Execution and Cumulative Audit around a strict separation among confirmed requirements, provisional predictions, and the mutable artifact with its provenance. The controller preserves the state triple (R_h, R_p, S) , but makes the distinction operational: compliance is judged only against active, evidence-supported requirements in R_h ; predictions in R_p are evaluated separately for utility, calibration, and waste; and external or consequential changes pass through action-specific authorization and commit gates. Each action carries a contract covering evidence, preconditions, expected effects, verification, side effects, reversibility, and commit criteria. The design favors preparation useful across several plausible futures, permits isolated branch-specific speculation only when its expected value is positive, and combines material-cycle checkpoints with event-triggered reviews. The nested 3-6-9 cadence is retained as an ablatable scheduling hypothesis, not a source of correctness. This paper defines the architecture and a controlled evaluation plan. It reports no live-model performance results and makes no claim that the proposed controller is superior until matched, independently evaluated experiments establish that fact.

1. From anticipation to evidence-gated execution

PECA v1 proposed coupling incremental instruction interpretation to changes in a persistent artifact and cumulative audits [9]. PECA v2 added an anticipatory layer: the agent predicts likely input, useful task-support actions, and relevant user preferences, then prepares work before the need is confirmed. That extension creates an important design tension. If a forecast is admitted into the same acceptance test as an explicit requirement, an agent can fail its own task by not satisfying an expectation it invented.

Consider a user who requests a minimal prototype. An agent predicts that tests, documentation, telemetry, and a configuration screen will eventually be useful. If these forecasts become part of its definition of completeness, it may build an elaborate system despite the request for minimality. The failure is not merely that the predictions were inaccurate. It is that predictions gained authority over the definition of success.

PECA v3 makes the governing rule explicit: *prediction may influence preparation; evidence determines compliance; authorization determines commitment*. The agent can explore plausible futures, but forecasts cannot create hard requirements. A prepared branch may be valuable, but it remains optional until supported by the actual task and validated under the appropriate commit gate.

This is a proposed control architecture, not a claim of a new model capability. The paper’s con-

tribution is an explicit separation of three questions that should not be conflated: is the artifact compliant, was anticipation useful, and may this change be committed to the real environment?

2. Relation to prior work

PECA v3 is situated among research on reasoning-action loops, iterative refinement, speculative execution, uncertainty calibration, and runtime auditing.

Action and critique. ReAct interleaves reasoning and actions with environmental feedback [1]. Self-Refine iteratively critiques and revises an output [2], and CRITIC uses tool interaction to check and amend model-generated content [3]. PECA shares iterative feedback but makes the distinction between confirmed requirements, predictions, and commit permissions part of the controller contract.

Confidence and personalization. Confidence estimates require calibration rather than trust in a model’s verbal probability [4]. PrefEval shows that inferring and following user preferences over long contexts is difficult for current LLMs [5]. These results motivate treating preference anticipation as fallible, scoped evidence rather than authority.

Speculation and audit placement. PASTE studies pattern-aware speculative tool execution to hide latency in agent serving [6]. Certificate-Gated Prefix Acceptance proposes a trusted verifier for gating speculative decisions under hard constraints [7]. BRA-Audit studies budget-aware runtime audit placement in multi-agent systems [8]. These

works show that anticipation and auditing are active design areas. PECA v3’s proposed distinction is a unified controller contract that separates hard-compliance evaluation from speculative utility while governing persistent artifacts and external commits. This distinction should be viewed as a hypothesis to test, not an established novelty or superiority claim.

3. State and authority boundaries

The operational state remains

$$\Sigma = (R_h, R_p, S),$$

where R_h is the hard requirement ledger, R_p is the predictive ledger, and S is the working artifact plus evidence, provenance, dependency information, and rollback history. An append-only event log can record observed input and tool results; counters and scheduler state can be derived from it. The triple is a compact authority model, not a claim that raw observations or runtime metadata cease to exist.

3.1. Hard requirement ledger R_h

Only active, evidence-supported instructions enter R_h . Each entry retains its source span, scope, status, dependencies, conflict state, and verification method. Interpretation must distinguish a user’s instruction from quoted material, examples, untrusted embedded text, and the agent’s own predictions. Later instructions may qualify or supersede earlier ones; the history remains auditable. Where instructions conflict or remain ambiguous, the system records an unresolved state or seeks clarification instead of silently inventing a reconciliation.

3.2. Predictive ledger R_p

The predictive ledger contains input-, task-, and preference-anticipation channels. Each forecast carries evidence, confidence calibrated for the relevant channel and horizon, a confirmation condition, a contradiction condition, expiry, proposed verification, estimated usefulness, and reversal cost. The model may maintain several plausible futures rather than a single top-one forecast. Robust preparations that remain useful across futures are preferred to branch-specific commitments.

3.3. Artifact state S

The artifact state stores the accepted work product and a provenance graph linking changes to their justification and verification. Speculative work should be isolated in disposable branches, drafts, or sandboxes. Rollback must account for dependent changes and external effects, not merely undo the

final file edit. Where full reversal is impossible, the action cannot be treated as safely speculative.

4. The evidence-gated six-phase cycle

Each meaningful input or task-state change triggers a flexible six-phase cycle. Phases may be no-ops when they have no useful work, but material state changes must still pass the checks appropriate to their risk.

1. **Sense.** Record the observed unit, tool output, or environment change with provenance.
2. **Interpret.** Reconcile the observed prefix into R_h . Do not promote a prediction merely because it is plausible or convenient.
3. **Anticipate.** Update R_p using current requirements and relevant authorized context. Keep prediction channels and future branches distinct.
4. **Select.** Prefer robust preparation with positive expected net value. Branch-specific speculation requires isolation, a rollback path, and no unmet authorization gate.
5. **Act.** Execute authorized hard-justified work or bounded speculative work. Consequential external side effects remain behind explicit permission gates.
6. **Audit and purify.** Run the hard compliance audit against R_h only. Separately score forecast utility, calibration, and wasted work. Repair hard failures; retain, revise, or discard speculative work without allowing a prediction-only failure to make the hard audit fail.

4.1. Action contracts and commit gates

Every material proposed action must carry a machine-checkable or reviewable contract: (i) the requirement or forecast that motivates it; (ii) preconditions and required permissions; (iii) the expected effect; (iv) the verification procedure; (v) direct and dependent side effects; (vi) a rollback or compensation plan; and (vii) criteria for committing the result. An executor should refuse an action when critical contract fields cannot be satisfied. This check belongs in the execution layer; it cannot depend solely on the model remembering to obey its own advice.

The governing invariant is:

$$\text{HardPass}(S) \iff \forall r \in R_h, \text{Verify}(r, S) = \text{pass},$$

subject to explicit unresolved states and applicable safety constraints. Forecasts are evaluated by a separate utility function, for example

$$U(a) = V_s - C_e - C_d - C_i - R_r,$$

where V_s denotes expected saved work, C_e execution cost, C_d discard cost, C_i integration cost, and R_r residual risk. A positive score is necessary but

not sufficient: safety, authority, and reversibility constraints remain hard gates.

5. Resonant scheduling without empty ceremony

The earlier 3-6-9 structure is retained as a scheduling hypothesis rather than a correctness rule. After three *material* micro-cycles, a meso-review examines prediction calibration, speculative waste, contradictory evidence, and whether anticipation should become more conservative. After nine material cycles, a macro-review examines goal coherence, stale assumptions, dependency integrity, unresolved requirements, and whether the controller should switch strategy. Counts refer to material cycles rather than characters or empty phases, so re-segmenting a prompt does not by itself change the review frequency.

Event-triggered reviews take precedence over the schedule. A safety incident, contradiction, failed critical verification, external-state surprise, or major scope change prompts immediate review. The fixed 3-6-9 policy must be compared with alternative fixed intervals and event-driven scheduling. Its structure is an organizing proposal; no special technical advantage follows from the numbers themselves.

6. Evaluation plan

The evaluation should compare direct execution, PECA v1, PECA v2, and PECA v3 using the same underlying model, tools, sampling configuration, task instances, and independent scoring criteria. Essential ablations include: no predictive layer; predictions computed but never acted upon; predicted requirements incorrectly merged into hard compliance; no action contracts; no rollback enforcement; fixed 3-6-9 versus event-triggered or alternative-interval reviews; and single-future versus multi-future anticipation.

Test both streaming and complete-prompt settings. Include late negation, requirement supersession, minimality conflicts, stale preference conflicts, prompt injection in quoted or retrieved content, exact-format tasks, destructive-action requests, code changes with indirect dependencies, and long-horizon tasks with several checkpoints. Streaming evaluation must withhold future text until it actually arrives. Completed-prompt evaluation must measure task preparation rather than pretend there is an unseen remainder of the prompt.

Primary metrics are explicit-requirement satisfaction, critical violations, unresolved conflicts, and regression rate. Safety metrics include unauthorized actions, failed commit-gate checks, and rollback completeness. Anticipation metrics include calibra-

tion, forecast precision and recall, retained speculative work, discarded-work fraction, and net saved latency. Efficiency metrics include tokens, tool calls, wall-clock latency, and cost. Report equal-budget and unconstrained-quality regimes, randomize task order, use repeated trials, bootstrap uncertainty at the task level, and blind evaluation wherever possible.

7. Limitations and threats to validity

PECA v3 is not validated by the controller simulations previously used during design exploration: those simulations rely on explicit assumed detection, repair, and speculation probabilities and cannot establish how a live language model behaves. No empirical performance result is claimed here. The proposed isolation rule can still fail if requirement interpretation is wrong, the verifier shares correlated errors with the executor, or the action contract omits a side effect. Formal guarantees require domain-specific trusted verifiers; ordinary language-model critique alone is not a proof.

Action contracts add implementation and integration cost. Strict gating can prevent useful work when permissions or rollback metadata are incomplete. A 3-6-9 schedule may waste compute or audit too infrequently; event-driven checkpoints may perform better. Forecasting may add little value in one-shot tasks, and personalized prediction can become stale or intrusive. These are reasons to test the components independently, not reasons to assume the architecture is effective.

8. Conclusion

PECA v3 refactors anticipatory execution around three distinct authorities: confirmed requirements define compliance, predictions guide preparation, and authorization rules determine whether changes can be committed. Its six-phase cycle combines semantic interpretation, multi-future anticipation, value-gated action selection, reversible execution, hard-only auditing, and a separate evaluation of predictive utility. Action contracts turn safety and rollback from stylistic advice into execution-time checks; nested reviews provide a testable scheduling policy with event-triggered overrides.

The proposal's promise is not that an agent predicts perfectly, nor that regular review intervals create correctness. It is that an agent may prepare ahead without allowing its own expectations to rewrite the task or bypass authorization. Whether that separation improves real task success enough to justify its cost remains an empirical question. The next step is a matched, blind, budget-controlled evaluation with component ablations and verifiable outcomes.

Drafting and reproducibility note

This manuscript was prepared with language-model assistance. It specifies a proposed controller and evaluation protocol, not completed empirical research. The author should verify all citations, review relevant prior art, revise the manuscript in their own voice, and follow the repository’s disclosure policies before publication.

References

- [1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629. <https://arxiv.org/abs/2210.03629>
- [2] Aman Madaan et al. 2023. Self-Refine: Iterative Refinement with Self-Feedback. arXiv:2303.17651. <https://arxiv.org/abs/2303.17651>
- [3] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/2305.11738>
- [4] Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koepl, Preslav Nakov, and Iryna Gurevych. 2024. A Survey of Confidence Estimation and Calibration in Large Language Models. In *Proceedings of NAACL-HLT 2024*, 6577–6595. <https://aclanthology.org/2024.naacl-long.366/>
- [5] Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. 2025. Do LLMs Recognize Your Preferences? Evaluating Personalized Preference Following in LLMs. In *International Conference on Learning Representations (ICLR)*. https://proceedings.iclr.cc/paper_files/paper/2025/hash/28a46044775d97a4efcbcf14e7f13209-Abstract-Conference.html
- [6] Yifan Sui, Han Zhao, Rui Ma, Zhiyuan He, Hao Wang, Jianxun Li, and Yuqing Yang. 2026. Act While Thinking: Accelerating LLM Agents via Pattern-Aware Speculative Tool Execution. arXiv:2603.18897. <https://arxiv.org/abs/2603.18897>
- [7] Chenyu Zhou, Qiliang Jiang, Shuning Wu, and Xu Zhou. 2026. Certified Speculative Execution for Untrusted AI Agents. arXiv:2606.31023. <https://arxiv.org/abs/2606.31023>
- [8] Kaixiang Wang, Yidan Lin, Jiong Lou, and Jie Li. 2026. BRA-Audit: Budgeted Runtime Auditing for LLM Multi-Agent Systems via Cumulative-Exposure Audit-Point Placement. arXiv:2608.14668. <https://arxiv.org/abs/2608.14668>
- [9] RM369. 2026. From Prompt-Level Goals to Prefix-Conditioned Execution (PECA v1). Life of Research. <https://lifeofresearch.com/preprints/137>