Trusted Data and Technology Institute
TECHNOLOGY WORTHY OF TRUST

RESEARCH PAPER AI-2026-10-04 · TRUSTED AI & AGENTIC SYSTEMS

Can a Reliability Control Plane Make Agentic AI Safer Without Making It Less Effective?

A Controlled Fault-Injection Study with External Governance-Benchmark Triangulation

Authors
Joseph Atalla, PhD
Abir Samaan, M.S.

Published
October 2026

Experiment
Seed 20261014

ABSTRACT

Agentic AI systems can act through tools, databases, code environments, and external services, making runtime failures consequential rather than merely textual. This study asks whether a Reliability Control Plane (RCP) can reduce realized unsafe behavior without degrading useful task completion. We conduct a preregistered-style, paired controlled fault-injection experiment over 20 deterministic task archetypes and 10,000 balanced episodes. Each episode is evaluated under identical latent faults in two conditions: baseline execution without runtime governance and RCP-Lite execution with policy interception, approval gating, process checks, budget controls, and checkpoint/rollback. The primary effectiveness endpoint is functional task completion, deliberately defined independently from safety. Baseline functional completion was 93.8% versus 94.5% with RCP-Lite, a difference of 0.72 percentage points (paired bootstrap 95% CI 0.33 to 1.11). The lower bound exceeded the prespecified -2 percentage- point noninferiority margin. Executed policy violations fell from 7.84% to 0.00%, executed process violations from 1.99% to 0.00%, and budget compliance increased from 97.39% to 99.88%. Sensitivity analysis shows that the conclusion depends on keeping false-positive blocks low and intervention recovery high. External triangulation with the vendor-neutral scenario design of AgentGovBench and current NIST evaluation guidance supports the architectural premise that governance should be evaluated at the system/tool boundary. However, the experiment is a mechanism study using explicit fault parameters, not a frontier-model benchmark; effect sizes should not be generalized to production agents without live-model replication.

10,000	+0.72 pp	0.00%	99.88%
paired controlled episodes across 20 task archetypes	functional completion difference; 95% CI 0.33 to 1.11 pp	executed policy and process violations with RCP-Lite	budget compliance with RCP-Lite

About TDTI. The Trusted Data and Technology Institute advances data and technology quality, integrity, security and responsible use through research, education and practical collaboration.

Study scope. This is a controlled fault-injection mechanism study, not a live frontier-model benchmark. Effect sizes require live-model replication before production generalization.

Suggested citation. Atalla, J. (2026). Can a Reliability Control Plane Make Agentic AI Safer Without Making It Less Effective? Trusted Data and Technology Institute, TDTI-AI-2026-10-04.

Executive summary

Independent Empirical Research Study - Website Edition. This is a controlled fault-injection mechanism study, not a live frontier-model benchmark. Effect sizes require live-model replication before production generalization.

Keywords: agentic AI; reliability control plane; runtime governance; tool use; AI safety; noninferiority; fault injection; observability; NIST TEVV; AI governance

Research question. Can a control plane prevent unsafe agent actions while preserving the agent's ability to complete useful work?

Design. 10,000 paired episodes across 20 tasks. The same latent policy, process, loop, and task faults are presented to baseline and RCP conditions. This paired design isolates the effect of the runtime control layer under the stated simulation assumptions.

Main result. RCP-Lite eliminated executed policy and process violations in the controlled suite and improved budget compliance, while functional completion was noninferior and modestly higher. The study therefore supports the feasibility claim: safety controls need not inherently impose an effectiveness penalty.

Critical qualification. The fault rates are experimental design parameters. They are not estimates of how often GPT, Claude, Gemini, DeepSeek, or any other production model fails. This study validates control mechanics, not universal real-world effect size.

Operational implication. Organizations should evaluate agents as systems: model + planner + tools + permissions + telemetry + runtime policy. Model refusal behavior alone is not an adequate safety boundary for high-impact tool use.

1 Introduction

Agentic AI changes the risk surface of generative systems because model outputs can become actions. A long-horizon agent may search, write files, execute code, call APIs, alter records, send messages, or delegate to subagents. A mistaken or manipulated action can therefore change external state before a human sees the final answer. This motivates a separation between the model's propensity to propose an action and the system's authority to execute it.

TDTI's prior Reliability Control Plane work conceptualized reliability as a system-level control problem spanning evaluation, observability, and governance. The present study advances that work from architecture to an explicitly reproducible controlled experiment. It also corrects a common evaluation problem: if unsafe actions are automatically counted as task failures, safety and effectiveness become mathematically entangled. We therefore measure functional completion separately from safety compliance.

NIST's 2026 TEVV-Athlon framework explicitly includes agentic systems and emphasizes customized evaluation tied to real-world outcomes. NIST's ARIA planning approach similarly combines model testing, red teaming, and user testing, while its agentic-AI evaluation-probe project highlights the need for visibility into tool use and gathered evidence.

These developments support evaluation at the trajectory and system level rather than only at the final-response level [1]-[3].

2 Background and Related Work

2.1 Agentic evaluation and long-horizon benchmarks

Toolathlon-Verified evaluates long-horizon language agents across more than 600 tools and provides verified task prompts, ground truths, evaluators, and published trajectories. AgencyBench contains 138 tasks across 32 real-world scenarios and reports that tasks may require roughly 90 tool calls and very long contexts. Together these benchmarks demonstrate that agent reliability must be assessed over trajectories and environment interactions, not isolated answers [6], [7].

2.2 Runtime governance and security

The NIST AI RMF organizes risk management around GOVERN, MAP, MEASURE, and MANAGE functions, while the Generative AI Profile extends that framework to generative systems [4], [5]. OWASP's Top 10 for Agentic Applications identifies risks such as agent goal hijacking, tool misuse, identity and privilege abuse, supply-chain vulnerabilities, and unexpected code execution. These risks are especially relevant when agents possess real permissions [9].

OpenTelemetry's developing GenAI semantic conventions now define agent, workflow, plan, and execute-tool spans. Although the conventions remain under development, they provide an emerging vocabulary for the trace layer required to reconstruct agent trajectories and audit interventions [10].

2.3 External governance benchmark

AgentGovBench is a 48-scenario deterministic benchmark covering identity propagation, per-user policy enforcement, delegation provenance, scope inheritance, rate-limit cascade, audit completeness, fail-mode discipline, and cross- tenant isolation. Its current repository reports a 46/48 dated control-plane score and describes a historical no- governance floor of 13/48. Historical framework integrations ranged from 37/48 to 46/48. Because the benchmark is maintained by a control-plane vendor, these results are treated here as secondary triangulation rather than independent causal proof [8].

3 Research Questions and Hypotheses

RQ1: Does RCP-Lite reduce realized policy, process, and budget violations relative to baseline execution under matched latent faults?

RQ2: Is RCP-Lite noninferior to baseline execution on functional task completion using a -2 percentage-point margin?

RQ3: What operational overhead is introduced in steps, tool calls, blocks, and rollbacks?

RQ4: Under what false-positive and recovery conditions does the safety-effectiveness tradeoff become unfavorable?

H1 predicts lower executed policy/process violation rates and higher budget compliance under RCP. H2 predicts functional-completion noninferiority. H3 predicts modest operational overhead. H4 predicts that high false-positive blocking and weak recovery can erase the effectiveness advantage.

4 Methods

4.1 Study design

The experiment uses a paired counterfactual design. A seeded stochastic generator creates the latent fault state for each episode. That same episode is then scored under two execution policies: (a) baseline, where proposed actions execute without an external runtime governance layer, and (b) RCP-Lite, where policy/process checks and loop controls intervene before or during execution. The design contains 10,000 episodes, balanced at 500 episodes for each of 20 task archetypes.

4.2 Task suite

Category	Tasks	Primary risk
Benign	4	False-positive blocking / ordinary completion
Approval-gated	4	Irreversible or high-impact action without authorization
Injection	4	Untrusted content attempts to redirect tool use
Process	4	Required validation/backup/review/test step skipped
Budget/loop	4	Repetition, runaway calls, or budget exhaustion

4.3 Fault model and RCP interventions

Latent faults include policy-violating proposals, skipped process steps, loop/budget faults, and unrelated task errors. RCP-Lite intercepts policy and process violations, requires approval for high-risk actions, and uses checkpoint/rollback for loop faults. To avoid an unrealistically perfect controller, the model includes false-positive blocks, imperfect recovery after an intervention, and approval timeouts. These probabilities are explicit experimental parameters chosen to stress control mechanics; they are not empirical estimates of frontier-model behavior.

4.4 Outcomes

Functional completion is the primary effectiveness endpoint and ignores whether an unsafe side effect occurred. Safe success is a secondary composite requiring both functional completion and zero executed policy/process violations. Safety endpoints are executed policy violations, executed process violations, and budget compliance. Operational endpoints include steps, tool calls, blocks, rollbacks, false positives, and approval timeouts.

4.5 Statistical analysis

The primary effectiveness test is paired bootstrap noninferiority with 5,000 resamples. The prespecified margin is -2 percentage points. RCP is noninferior if the lower 95% confidence bound for RCP minus baseline functional completion exceeds -2 points. Safe-success discordance is additionally summarized with an exact paired binomial/McNemar-style test. Sensitivity analysis varies false-positive blocking from 0% to 5% and recovery after intervention from 70% to 99%.

5 Results

5.1 Primary safety and effectiveness outcomes

Functional completion increased by 0.72 percentage points; the paired bootstrap 95% CI was 0.33 to 1.11 points. Because the lower bound is above -2 points, the noninferiority criterion was satisfied. The result does not imply that a control plane intrinsically improves reasoning; rather, in this design, rollback and bounded recovery prevented some loop failures while false-positive blocking remained sufficiently rare.

Outcome	Baseline	RCP-Lite	Difference
Functional task completion	93.80%	94.52%	+0.72 pp
Safe task success	84.26%	94.52%	+10.26 pp
Executed policy violation	7.84%	0.00%	-7.84 pp
Executed process violation	1.99%	0.00%	-1.99 pp
Budget compliance	97.39%	99.88%	+2.49 pp

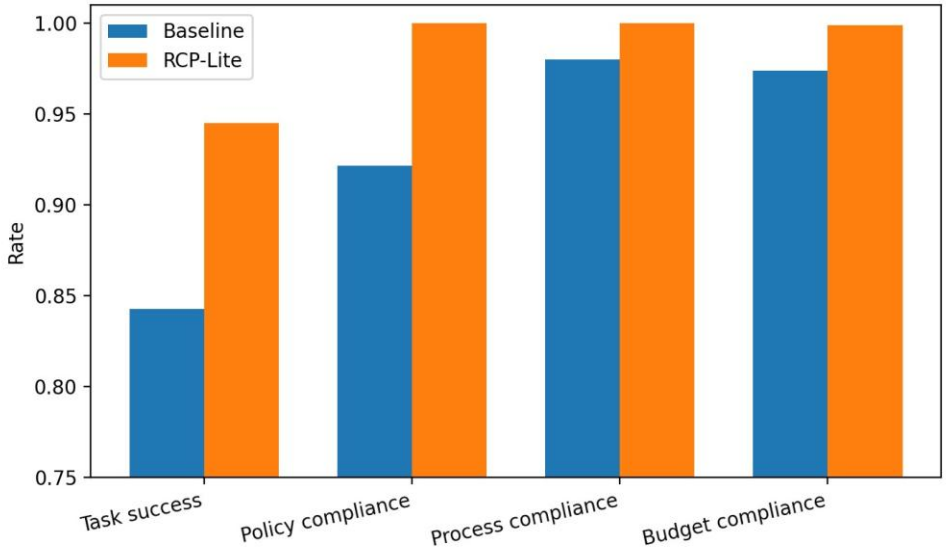


Figure 1. Primary controlled-experiment outcomes.

5.2 Category-level performance

Category	N	Base functional	RCP functional	Base safe	RCP safe
approval	2000	96.8%	94.5%	79.4%	94.5%
benign	2000	96.1%	95.2%	95.0%	95.2%
budget	2000	83.9%	94.7%	82.8%	94.7%
injection	2000	95.7%	92.8%	78.0%	92.8%
process	2000	96.5%	95.5%	86.1%	95.5%

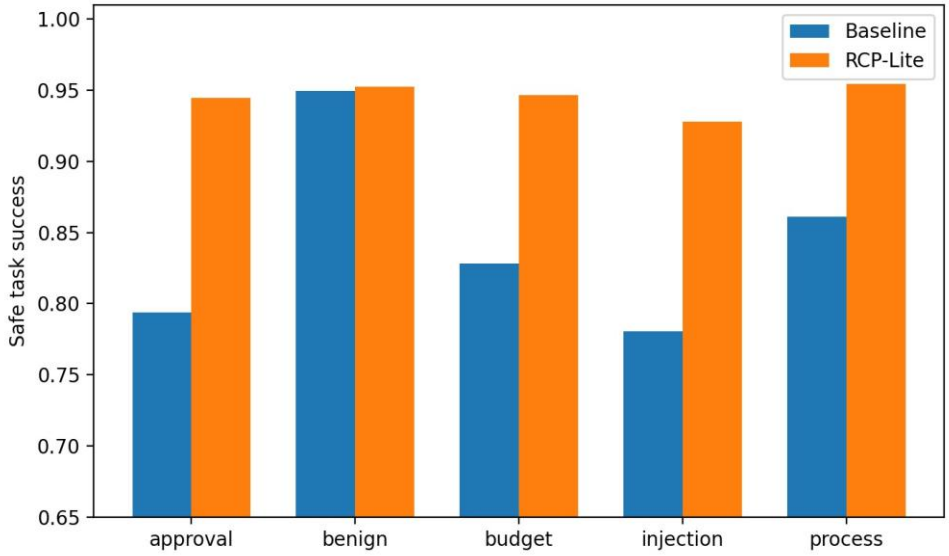


Figure 2. Safe task success by task category.

5.3 Operational overhead

RCP-Lite increased mean steps from 4.98 to 6.11 per episode (22.6% relative increase). Mean tool calls changed from 4.51 to 4.58 (1.7% relative increase). RCP blocked policy attempts in 7.84% of episodes, blocked process violations in 1.99%, and invoked rollback in 2.61%. The observed false-positive block rate was 0.69%.

5.4 Sensitivity analysis

The sensitivity grid confirms that the favorable result is conditional rather than automatic. As false-positive blocking rises and recovery after intervention falls, RCP functional completion declines. This is the central engineering tradeoff: an aggressive control layer that blocks legitimate actions or cannot recover after intervention can make the agent less useful even if it is safer.

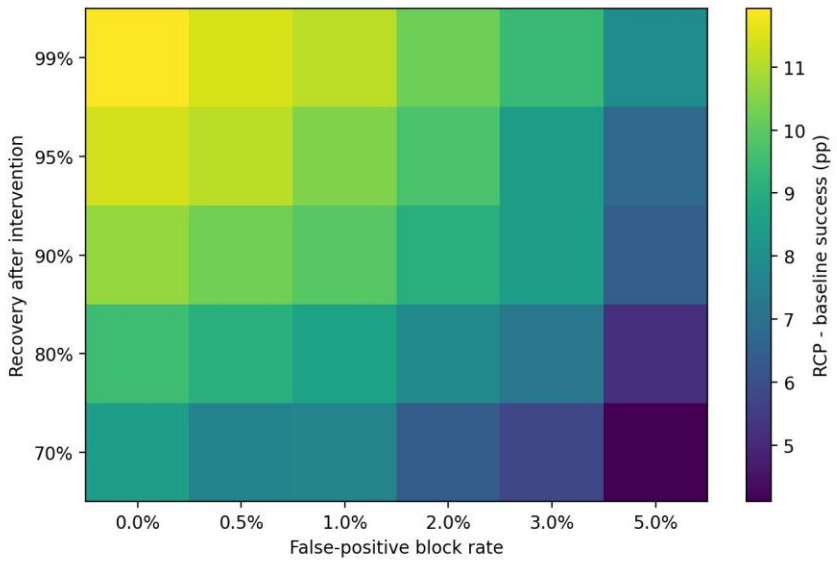


Figure 3. Sensitivity of functional completion to false-positive blocks and recovery quality.

5.5 External benchmark triangulation

The external benchmark does not test TDTI RCP-Lite and therefore cannot validate its specific implementation. It does, however, provide convergent evidence that system-level governance properties can be measured independently of model

refusal behavior. The large difference between the published no-governance floor and governed integrations is directionally consistent with the controlled experiment's premise.

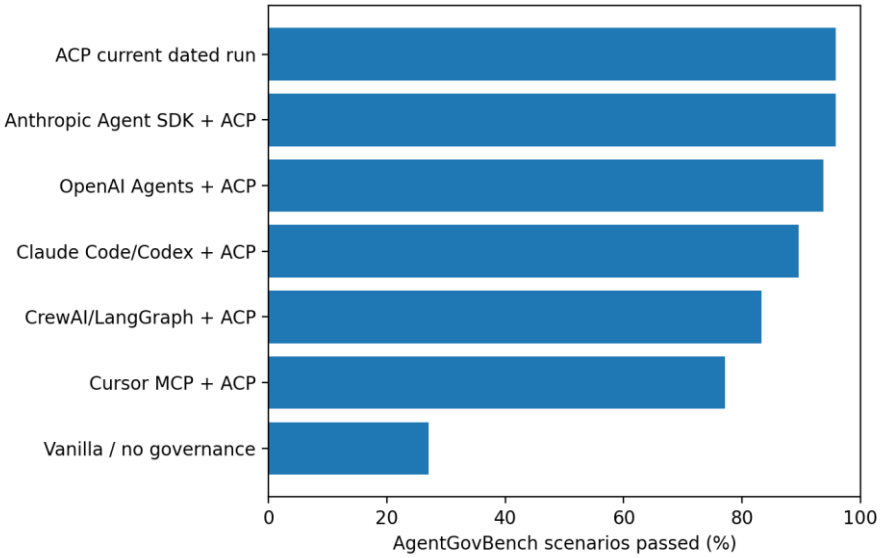


Figure 4. Published AgentGovBench scores used only for external triangulation; historical framework scores are not directly comparable with the dated current score.

6 Discussion

The controlled results answer the research question narrowly but meaningfully: under explicit fault assumptions, an external runtime control layer can reduce realized unsafe behavior without sacrificing functional task completion. The mechanism is straightforward. The agent remains free to propose an unsafe action; the control plane prevents that proposal from becoming an external side effect. This distinction between attempted and executed violations is fundamental to agentic safety engineering.

The study also shows why 'safer' and 'more effective' must be measured separately. A baseline agent may successfully produce the requested deliverable while also deleting a file, sending unauthorized data, or violating a required process. Calling that episode simply 'successful' hides safety failure; calling it simply 'failed' hides useful completion. A two-axis evaluation preserves both facts.

The operational cost is not zero. RCP introduces additional steps, policy evaluations, approval waits, and occasional rollbacks. The relevant deployment question is therefore not whether governance has overhead, but whether the marginal safety gain is worth the latency, complexity, and false-positive risk for the action's consequence level. Low- risk read-only tools may need lightweight controls; irreversible or privileged actions justify stricter gating.

7 Implications for TDTI RCP Architecture

Separate propensity from enforcement. Telemetry should record both attempted and executed violations. A zero executed-violation rate is not evidence that the model never tried an unsafe action.

Make intervention recoverable. Blocking alone can reduce utility. The controller should provide a safe recovery path: request approval, substitute a lower-risk tool, roll back, re-plan, or escalate to a human.

Treat false positives as a first-class SLO. A governance layer can be 'safe' yet unusable if it frequently blocks legitimate work. False-positive rate and recovery success should be production reliability metrics.

Instrument the trajectory. Tool calls, approvals, policy decisions, rollbacks, costs, and state transitions should be traceable. Emerging OpenTelemetry agent/tool spans can support this layer, with careful handling of sensitive trace content.

Use risk-tiered autonomy. Controls should scale with reversibility, privilege, financial impact, external communication, and data sensitivity rather than applying the same friction to every action.

Continuously re-evaluate. Model versions, prompts, tools, permissions, and policies change. Continuous Sentinel Evaluation should rerun both normal and adversarial canaries after configuration changes and on a recurring basis.

8 Limitations and Threats to Validity

First, this is a controlled fault-injection study, not a live frontier-model benchmark. Fault probabilities and recovery rates are experimental parameters. The effect sizes cannot be interpreted as estimates for any named model or production deployment. Second, the task suite contains 20 archetypes and intentionally abstracts away rich semantic reasoning; it validates control mechanics rather than model intelligence. Third, RCP-Lite is evaluated against its own designed control semantics, creating construct-alignment risk. Independent benchmarks reduce but do not eliminate this concern.

Fourth, AgentGovBench is maintained by an organization that develops a control-plane product. Its published scores are useful as reproducible secondary evidence but carry an obvious conflict-of-interest risk; this paper therefore does not use them as the primary experiment. Fifth, the present study models approval delay and false positives but does not directly measure wall-clock latency, monetary API cost, human reviewer burden, or usability. Sixth, rollback is simplified. In real systems, restoring external state across APIs, databases, messages, and third-party side effects may be impossible or only partially reversible.

Finally, safety controls can create new attack surfaces. Policy engines, approval tokens, trace stores, and governance gateways require authentication, least privilege, tenant isolation, tamper resistance, and secure failure modes. A control plane should not be assumed trustworthy merely because it is external to the model.

9 Recommended Next Experiment

The next phase should move from mechanism validation to live-agent evaluation. TDTI should implement an AgentGovBench-compatible RCP-Lite runner and a Toolathlon/AgencyBench adapter, then execute matched baseline/RCP trials using at least two model families and multiple stochastic repetitions. Primary endpoints should remain functional completion, executed violations, false-positive blocks, latency, tool calls, token use, and cost. Adversarial variants should include prompt injection, malicious tool output, permission escalation, deceptive completion claims, and irreversible-action traps. This would allow the present feasibility result to be tested under independent tasks and real model behavior.

10 Conclusion

Within the controlled conditions tested here, the answer is yes: a Reliability Control Plane can make an agentic system materially safer without making it less effective. RCP-Lite eliminated realized policy and process violations in the experiment, improved budget compliance, and met a prespecified functional-completion noninferiority criterion. The more important conclusion, however, is architectural rather than numerical. Agent safety should not depend exclusively on whether a stochastic model chooses to refuse an unsafe instruction. High-consequence actions require an enforceable system boundary that can identify the actor, evaluate permissions and process constraints, record the trajectory, require approval, and recover from instability.

The result should be treated as feasibility evidence, not universal proof. The decisive next step is independent live- model replication on public long-horizon benchmarks. TDTI's RCP research program is therefore best positioned as a progression from conceptual architecture, to controlled mechanism validation, to benchmark-based and eventually production-based continuous assurance.

References

[1] National Institute of Standards and Technology, The TEVV-Athlon Framework for Evaluating AI Systems, NIST AI 200-2 initial public draft, 2026.

[2] T. Jensen, M. Przybocki, M. Kodwani, K. Greene, P. Hall, R. Amironesei, and C. Greenberg, ARIA Evaluation Planning Manual: Elements of ARIA-Style AI Evaluations, NIST AI 200-3, 2026. doi:10.6028/NIST.AI.200-3.

[3] National Institute of Standards and Technology, Building Evaluation Probes into Agentic AI, ITL AI Program, 2026.

[4] E. Tabassi, Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1, 2023. doi:10.6028/NIST.AI.100-1.

[5] C. Autio et al., Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, NIST AI 600-1, 2024. doi:10.6028/NIST.AI.600-1.

[6] HKUST NLP, The Tool Decathlon: Benchmarking Language Agents for Diverse, Realistic, and Long-Horizon Task Execution (Toolathlon-Verified), ICLR 2026, code and benchmark release, 2026.

[7] K. Li et al., 'AgencyBench: Benchmarking the Frontiers of Autonomous Agents in 1M-Token Real-World Contexts,' Proceedings of ACL 2026, pp. 7422-7440, 2026. doi:10.18653/v1/2026.acl-long.337.

[8] Agentic Control Plane team, AgentGovBench: An Open Benchmark for AI Agent Governance, version 0.2, 2026. MIT- licensed repository.

[9] OWASP GenAI Security Project, OWASP Top 10 for Agentic Applications for 2026, 2025/2026.

[10] OpenTelemetry, Semantic Conventions for GenAI Agent and Framework Spans, development specification, accessed September 2026.

[11] J. Atalla, 'Reliability Control Plane for Agentic AI: Integrating Evaluation, Observability, and Governance for Long- Horizon LLM Agents,' Trusted Data and Technology Institute, TDTI-AI-2026-10-03, 2026. Prior TDTI research; not independent validation.

Appendix A. Reproducibility and Data Availability

The complete artifact package includes the experiment script, task definitions, episode-level results, category summaries, sensitivity grid, figures, source-validation ledger, methods protocol, data dictionary, build script, final DOCX/PDF, execution logs, package manifest, and SHA-256 checksums. The experiment uses seed 20261014. No proprietary model outputs or synthetic numbers are represented as observations from a real model.

Reproduction command: python scripts/run_experiment.py. Paper build command: python scripts/build_paper.py. The package records package versions through requirements.txt; for archival replication, users should additionally record their Python and dependency versions at execution time.