Trusted Data and Technology Institute
TECHNOLOGY WORTHY OF TRUST

RESEARCH PAPER AI-2026-10-03 · TRUSTED AI & AGENTIC SYSTEMS

Reliability Control Plane for Agentic AI

Integrating Evaluation, Observability, and Governance for Long-Horizon LLM Agents

Author
Joseph Atalla, PhD

Published
October 2026

Edition
Independent research paper

ABSTRACT

Agentic AI systems, defined here as LLM-centered systems augmented with tools, memory, and multi-step planning, enable long-horizon, goal-directed task execution. However, stochastic execution, trajectory-dependent failures, and tool-mediated security risks challenge conventional software testing and monitoring. This paper provides a scoping review and synthesis of three reliability pillars for agentic systems: (i) evaluation methods and benchmarks for interactive and long-horizon agents, (ii) observability standards and tracing primitives that enable trajectory reconstruction and diagnosis, and (iii) governance frameworks and regulatory drivers that shape deployment and post-deployment monitoring. Building on agent benchmarks, emerging OpenTelemetry semantic conventions for GenAI telemetry, and governance frameworks including NIST AI RMF and the NIST GenAI Profile, we propose a unified Reliability Control Plane (RCP) architecture that integrates trace-centric evaluation, standardized telemetry, and runtime policy enforcement into a continuous assurance loop. We additionally introduce RCP-Lite, a lightweight reference implementation, and a 20-task outcome-grounded micro-suite. In a stochastic proxy evaluation (N=1000 balanced episodes), RCP-Lite improved task success (94.2%→100%) and eliminated executed policy, process, and budget violations while leaving the agent's attempted policy-violation rate unchanged (8.6%). Finally, we propose Continuous Sentinel Evaluation, a post-deployment canary-testing pattern that repeatedly runs governance-aware tests to detect drift and to bound violation probabilities over time.

1,000	+5.8 pp	0	50
balanced stochastic proxy episodes across 20 tasks	task-success improvement in the stochastic proxy evaluation (94.2% to 100%)	executed policy, process and budget violations with RCP-Lite in the proxy suite	hosted DeepSeek episodes; both conditions achieved 100% task success

About TDTI. Trusted Data and Technology Institute conducts independent applied research on evaluation, observability, governance and continuous assurance for long-horizon agentic AI systems.
Research scope. This paper combines a scoping review, an RCP architecture, a stochastic proxy evaluation and a small hosted-agent evaluation.
Suggested citation. Atalla, J. (2026). 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.

I. INTRODUCTION

Index Terms-agentic AI; LLM agents; reliability engineering; evaluation; benchmarks; observability; tracing; OpenTelemetry; governance; NIST AI RMF; prompt injection

Agentic AI is increasingly used to move from single-turn text generation to goal-directed systems that plan, act via tools, and iterate over multiple steps. The emerging landscape includes a wide variation in how 'agents' are defined and operationalized, which complicates both evaluation and governance; recent synthesis efforts emphasize the need for clearer conceptual foundations and measurement practices. [1]

A growing body of work argues that agent reliability cannot be reduced to a single response quality score. In tool-using systems, the unit of behavior is a trajectory: a sequence of model invocations, tool actions, and environment observations whose errors can compound over time. Surveys of LLM-agent evaluation emphasize long-horizon interaction, tool execution, and safety constraints as key open challenges for rigorous assessment. [2]

Benchmarking efforts reflect this shift from static prompts to interactive environments. AgentBench evaluates LLMs as agents across multiple interactive settings. [3] OdysseyBench targets long-horizon office application workflows and highlights that atomic tasks often miss realistic multi-step dependencies.

[4] Related work explores end-to-end experimentation workflows (MLAgentBench), mobile and GUI agents (Mobile-Bench), and long-horizon partially observable environments (UltraHorizon), collectively reinforcing that long-horizon reliability is a distinct engineering problem. [5]-[7]

In parallel, work on measuring agent autonomy notes that provider-side logs often do not naturally 'sessionize' multiple calls into a single agent run without developer instrumentation, complicating empirical study and root-cause analysis. [8] Practical agent-building guidance also emphasizes modular patterns (planner, tool selection, memory, and feedback loops) rather than monolithic 'one-shot' prompting, which further motivates a system-level reliability approach. [9]

This paper proposes that reliability engineering for agentic systems is best treated as a control-plane problem. We define a Reliability Control Plane (RCP) that integrates three pillars: evaluation (trace-aware, outcome-grounded testing), observability (standardized telemetry for trajectory reconstruction), and governance (runtime policy and audit controls). The goal is to provide a framework-agnostic blueprint that enables continuous assurance across development, deployment, and post-deployment monitoring. We additionally provide a lightweight reference implementation (RCP-Lite), empirical validation on an outcome-grounded micro-suite, and an operational pattern for Continuous Sentinel Evaluation to keep deployed agents in check as models, prompts, and tools evolve.

The main contributions of this work are:

1) A scoping review and synthesis of agent evaluation, observability standards, and governance frameworks relevant to production agentic AI.

2) A trajectory-level reliability model separating outcome success, process conformance, policy compliance, and operational constraints.

3) A failure-mode taxonomy for tool-using, long-horizon agents mapped to control-plane mitigations.

4) A reference Reliability Control Plane architecture and minimal interfaces for trace-aware evals, telemetry, and runtime policy enforcement.

5) A reliability metric set and service-level objectives (SLOs) that separate attempted versus executed policy/process violations and quantify intervention overhead (blocks, rollbacks, loops).

6) A lightweight RCP-Lite reference implementation and a 20-task outcome-grounded micro-suite with balanced scheduling, including empirical results for a stochastic proxy agent (N=1000) and a hosted LLM agent (N=50).

7) Operational guidance for Continuous Sentinel Evaluation (CSE): continuously running governance-aware canary tests post-deployment to detect drift, enforce autonomy tiering, and bound violation probabilities over time.

II. REVIEW METHODOLOGY

This paper follows a scoping review methodology to map a rapidly evolving technical landscape and synthesize actionable engineering guidance. Scoping reviews are appropriate when key concepts are still being defined and when the objective is to identify categories of evidence and research gaps rather than to estimate effect sizes. [10], [11]

We use the PRISMA-ScR reporting principles as a guide for transparently describing the scope, inclusion criteria, and synthesis approach. [12] Sources are grouped into three reliability pillars: (i) evaluation and benchmarking of LLM agents; (ii) observability standards, semantic telemetry, and tracing; and (iii) governance and risk management frameworks relevant to deployment and monitoring.

Inclusion criteria were: (a) peer-reviewed benchmarks or surveys proposing evaluation taxonomies for interactive or long-horizon agents; (b) standards specifications or authoritative documentation defining telemetry semantics for GenAI or agentic systems; and (c) governance, risk management, or security frameworks with explicit lifecycle guidance applicable to AI systems. The synthesis extracts design requirements and organizes them into a unified control-plane reference architecture.

III. DEFINITIONS AND RELIABILITY MODEL

A. Agentic system model

We model an agentic system as a tuple:

$$A = (M, \Pi, T, S, E) \quad (1)$$

where M is the model service, Π is the agent policy (planner/router), T is the set of tools (APIs, code execution, databases), S is internal state/memory, and E is the external environment. This abstraction is consistent with the evaluation literature, which treats tool use and environmental interaction as defining properties of agentic behavior [2].

B. Trajectory and evaluation unit

A run generates a trajectory:

$$\tau = \{(x_t , a_t , o_t)\} \text{ for } t = 1, \ldots, T \quad (2)$$

where x_t is context/state, a_t is an action (including tool calls), and o_t is an observation (tool output or environment feedback). In long-horizon benchmarks, correctness depends on both the trajectory and the final output, motivating trace-aware evaluation and debugging [4].

C. Multi-objective reliability

We define agentic reliability as satisfying an explicit specification over a task distribution subject to constraints: (i) outcome success (final environment state or artifacts), (ii) process conformance (required workflow/tool constraints), (iii) policy compliance (no forbidden actions or data access), and (iv) operational bounds (latency and cost budgets). This definition aligns with the evaluation survey's emphasis on reliability and safety beyond single-turn response quality. [2]

IV. EVALUATION AND BENCHMARKING

A. Benchmark landscape for interactive and long-horizon agents

Agent benchmarks have progressed from static QA-style datasets to interactive environments where agents must choose actions, invoke tools, and manage state. AgentBench provides a multi-environment benchmark for LLM-as-agent evaluation. [3] OdysseyBench focuses on complex office application workflows and emphasizes long-horizon dependencies and artifact-based outcomes. [4] MLAgentBench targets end-to-end ML experimentation workflows that naturally support outcome verification. [5] Mobile-Bench evaluates mobile and GUI agents, highlighting environment coupling and interface constraints. [6] UltraHorizon introduces long-horizon partially observable settings that stress planning and memory. [7]

TABLE I BENCHMARKS COMMONLY USED TO STUDY AGENT RELIABILITY

Benchmark	Primary setting	Reliability signal	Strength/limitation
AgentBench [3]	Interactive multi-environment tasks	Task success and interaction outcomes	Broad coverage; limited realism for domain-specific enterprise workflows
OdysseyBench [4]	Office applications (documents, spreadsheets, email-like tasks)	Artifact correctness over long-horizon workflows	High realism; domain-scoped to office app scenarios
MLAgentBench [5]	ML experimentation pipelines	End-to-end experiment outputs and results	Outcome-grounded; narrower task domain
Mobile-Bench [6]	Mobile/GUI agent tasks	Task completion in a GUI environment	Tests UI/tool coupling; instrumentation and reproducibility challenges
UltraHorizon [7]	Long-horizon partially observable environments	Long-run goal achievement under partial observability	Stresses planning/memory; benchmarking maturity evolving

B. Evals as engineering infrastructure

Recent engineering guidance emphasizes evaluations as structured tests with explicit tasks, graders, and repeatable execution, rather than ad hoc prompt comparison. [13] For agentic systems, evals should be outcome-grounded when possible (e.g., unit tests, file diffs, database assertions) and should score both outcomes and trajectory constraints (e.g., required tools, approval steps). [13]

Platform documentation reinforces continuous evaluation throughout the development-to-deployment loop. OpenAI provides an Evals API for programmatic evaluation, guidance on evaluation best practices, and patterns for systematically testing agent skills. [14]-[16] The OpenAI Evals repository further illustrates reusable evaluation harnesses and graders. [17]

A practical evaluation stack for production agents includes: (i) offline regression suites used as release gates; (ii) scenario stress tests including adversarial and edge-case tasks; (iii) shadow evaluation on production-like inputs without user impact; and (iv) online monitoring signals derived from telemetry (e.g., tool error rates, policy blocks, loop detection). Multi-trial execution is recommended when stochastic variance is material to reliability.

In addition to offline suites and shadow evaluation, we introduce Continuous Sentinel Evaluation (CSE): scheduled execution of a compact, high-signal task battery against the current production agent configuration. CSE functions as synthetic monitoring for agentic systems, enabling early drift detection and confidence-bounded estimates of policy/process violation rates using the same trace-aware graders and policy engine as production.

V. FAILURE MODE TAXONOMY

Benchmark evidence and evaluation surveys indicate that agent failures are often trajectory-dependent and fall into recurring categories. [2]-[7] We organize failures into five classes to support systematic diagnosis and mitigation design:

TABLE II FAILURE MODES AND RELIABILITY CONTROL PLANE MITIGATIONS

Failure class	Typical trace symptoms	Eval strategy	Control-plane mitigations
Outcome failure	Final artifacts/state incorrect; late-stage divergence	Outcome-grounded checks (tests, diffs)	Artifact verification; regression suites; canary gates
Process failure	Wrong tool sequence; missing required steps; thrashing	Trace-aware constraints; tool-call rules	Tool routing policies; loop detection; guardrail events
Memory/context failure	Contradictory assumptions; retrieval mismatch; state drift	Long-horizon tasks; repeated-trial stability tests	Sessionized traces; memory instrumentation; replay tooling
Safety/security failure	Tool inputs contain injected instructions; unsafe action attempt	Adversarial scenarios; red-team evals	Policy gating; input validation; sandboxing
Governance/ops failure	No audit trail; approvals bypassed; SLO budget violations	Policy compliance scoring; cost/latency bounds	Centralized policy engine; audit logs; SLO monitors

The taxonomy is intended to be operational: each failure class should have measurable telemetry signals, explicit evaluation coverage, and a mapped set of runtime controls. This mapping motivates the unified Reliability Control Plane described next.

VI. RELIABILITY CONTROL PLANE ARCHITECTURE

We define a Reliability Control Plane (RCP) as a set of cross-cutting services that provide consistent measurement, monitoring, and policy enforcement across heterogeneous agent frameworks and tool ecosystems. The agent runtime (data plane) executes the task, while the RCP collects telemetry, runs evaluations, and enforces governance constraints.

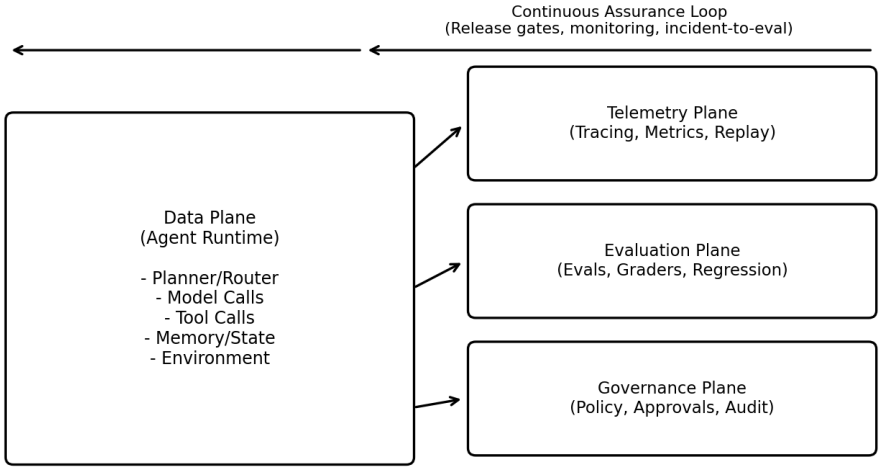


Figure 1. Reliability Control Plane (RCP) reference architecture (conceptual).

A. Control-plane components

The RCP contains three core planes: (i) a Telemetry Plane that stores sessionized traces and operational metrics; (ii) an Evaluation Plane that orchestrates regression suites, scenario tests, and trace-aware graders; and (iii) a Governance Plane that implements permissions, approvals, and audit logging for tool use and data access. A reliability operations loop ties these planes together through release gates and incident-to-eval feedback.

B. Design requirements

To be practically deployable, an RCP should be: (1) framework-agnostic, supporting multiple agent runtimes; (2) trace-centric, enabling trajectory reconstruction and replay; (3) policy-aware, logging and enforcing decisions at runtime; and (4) evaluation-driven, converting failures into permanent regression tests.

VII. TRACE-AWARE EVALUATION PIPELINE

A distinguishing requirement for agent reliability is that evaluation should consume both the outcome and the execution trace. Trace-aware grading enables enforcement of workflow constraints (e.g., required tools, approvals) and provides evidence for root-cause analysis.

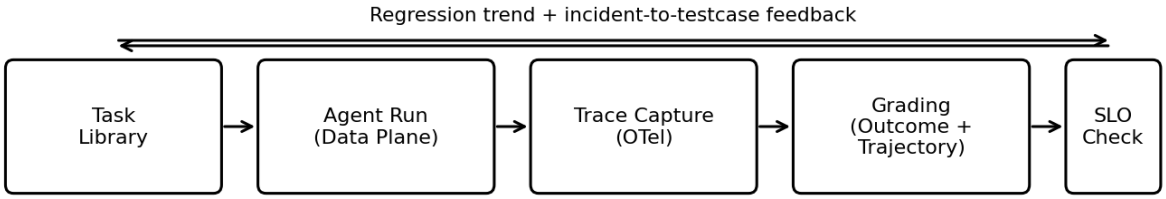


Figure 2. Trace-aware evaluation pipeline with regression feedback loop.

In the pipeline, each task run emits a trace that is stored for replay and linked to artifacts and grades, and aggregated metrics support release gating and post-deployment monitoring. Incidents are triaged into new test cases, strengthening the regression suite over time.

VIII. OBSERVABILITY PLANE

Trajectory-level debugging requires instrumentation that captures model calls, tool calls, and agent-step boundaries with correlation identifiers. Agent runtimes are increasingly shipping built-in tracing; for example, the OpenAI Agents SDK exposes traces and spans for generations, tool calls, handoffs, and guardrail events. [18]

To enable cross-framework interoperability, OpenTelemetry has introduced semantic conventions for generative AI systems that standardize how to represent model, agent, event, and metric data. [19] The conventions include explicit agent and framework spans suitable for representing agent runs and steps. [20] Complementary event and metric definitions support operational monitoring for latency, token usage, and error signals. [21]

Provider- and protocol-specific conventions further support consistent instrumentation. OpenTelemetry defines semantic conventions for OpenAI client operations, enabling consistent telemetry for model requests and responses. [22] For tool ecosystems that use the Model Context Protocol (MCP), OpenTelemetry also provides MCP-specific conventions to capture domain-relevant context beyond generic RPC telemetry. [23]

A. Minimal telemetry schema

A minimal viable observability schema should include: (i) a session or run identifier; (ii) model call spans with provider, model, latency, and token usage; (iii) tool call spans with tool name and redaction markers for sensitive arguments; (iv) agent step events (plan, act, observe); and (v) policy decision events (allow, block, escalate) bound to the same trace.

B. Privacy and data handling

Because traces may contain prompts, retrieved content, and tool outputs, organizations should treat trace data as potentially sensitive and apply access control, minimization, and retention policies. Telemetry design should support redaction and structured logging to enable reliability analysis without indiscriminately capturing sensitive content.

IX. GOVERNANCE PLANE

Governance frameworks increasingly require lifecycle risk management, continuous measurement, and accountability for AI systems. The NIST AI Risk Management Framework (AI RMF 1.0) defines a lifecycle structure (GOVERN, MAP, MEASURE, MANAGE) and identifies 'valid and reliable' as a key trustworthiness characteristic. [24] The NIST Generative AI Profile maps these functions to GenAI-specific risks and emphasizes post-deployment monitoring and management. [25]

Agentic-specific governance guidance further motivates the use of runtime controls. The IMDA Model AI Governance Framework for Agentic AI highlights planning, tool use, and immature tool protocols as risk multipliers and recommends baseline testing and technical controls across the agent lifecycle.

[26] ISO/IEC 42001 provides requirements for an AI management system and continuous improvement processes, reinforcing the need for measurable assurance artifacts. [27]

Regulatory frameworks such as the EU AI Act impose obligations on certain AI systems, including requirements for post-deployment monitoring and documentation. [28] While applicability depends on system classification and jurisdiction, these obligations increase the practical importance of audit logs, monitoring plans, and evidence of ongoing performance management.

Security guidance identifies prompt injection as a primary risk category for LLM applications. [29], [30] In agentic settings, prompt injection can escalate from incorrect text to harmful tool-mediated actions, motivating governance controls that enforce least privilege, approval gating for irreversible actions, input validation for tool payloads, and sandboxing for code execution.

A. Autonomy tiering and action gating

A practical governance mechanism is autonomy tiering: classify tools and actions by risk (reversibility, financial impact, data sensitivity) and bind each tier to required approvals and logging. High-risk actions (payments, deletions, external communications) should require explicit human confirmation or redundant supervisory review (e.g., a secondary 'supervisor' agent) before execution, while lower-risk actions may proceed with automated policy checks and comprehensive audit logging.

X. RELIABILITY METRICS AND SLOS

Reliability metrics should reflect outcome correctness, policy compliance, process conformance, and operational bounds, and should separate (i) agent behavior (attempted violations) from (ii) realized system behavior after enforcement (executed violations and blocks). For episode i, let $S_i \in \{0,1\}$ denote task success; let A_i^P, V_i^P, and B_i^P indicate whether any policy-violating action was respectively attempted, executed, or blocked; and define analogous A_i^C, V_i^C, and B_i^C for process constraints. Let K_i denote tool-call count, L_i denote step count, D_i denote loop detections, and R_i denote rollbacks.

$$TSR = (1/n) \Sigma_{i=1}^{n} S_i \quad (3)$$

$$PCR_{exec} = 1 - (1/n) \Sigma_{i=1}^{n} V_i^P \quad (4)$$

$$PCS_{exec} = 1 - (1/n) \Sigma_{i=1}^{n} V_i^C \quad (5)$$

$$VarSR = Var(S_i) \text{ for } i = 1, \ldots, n \quad (6)$$

To quantify agent propensity versus enforcement effectiveness, we additionally report attempt compliance rates $PCR_{attempt} = 1 - (1/n) \Sigma_{i=1}^{n} A_i^P$ and $PCS_{attempt} = 1 - (1/n) \Sigma_{i=1}^{n} A_i^C$, as well as policy/process block rates $PBR = (1/n) \Sigma_{i=1}^{n} B_i^P$ and $CBR = (1/n) \Sigma_{i=1}^{n} B_i^C$. Operational SLOs can be expressed as budget compliance $BCR = (1/n) \Sigma_{i=1}^{n} 1[K_i \le K_{max} \wedge L_i \le L_{max}]$, together with expected resource usage $E[K_i]$ and $E[L_i]$. Loop and intervention signals support early detection of thrashing: loop incidence $LIR = (1/n) \Sigma_{i=1}^{n} 1[D_i > 0]$, rollback incidence $RIR = (1/n) \Sigma_{i=1}^{n} 1[R_i > 0]$, and mean loop detections $E[D_i]$.

In Continuous Sentinel Evaluation, repeated execution of the same battery enables confidence-bounded estimates of rare-event probabilities. For example, observing zero executed policy violations across N sentinel episodes implies an approximate 95% upper bound of 3/N on the per-episode executed-violation probability under the sentinel distribution.

XI. IMPLEMENTATION BLUEPRINT

A minimal path to implementing an RCP can be staged: (1) instrument agent runs with sessionized traces (model, tool, and step spans); (2) establish a baseline eval suite with outcome-grounded grading and trace-aware constraints; (3) introduce a centralized policy engine for tool permissions and approvals; and (4) operationalize release gates and incident-to-eval feedback loops. This staged approach supports incremental maturity without requiring a complete platform rebuild.

A. RCP-Lite reference implementation

To ground the architecture in executable artifacts, we implemented RCP-Lite, a lightweight wrapper around an agent runtime that (i) intercepts proposed tool calls, (ii) emits trace events, (iii) evaluates policy and process constraints, and (iv) applies runtime interventions (allow, block, require approval, checkpoint/rollback). RCP-Lite supports autonomy tiering by labeling tools/actions by risk and binding high-risk actions (e.g., irreversible deletion) to explicit approvals and additional logging.

B. Checkpoint and rollback semantics

Checkpoint/rollback must restore both the environment state and the agent's internal state S (e.g., memory, plan pointer, message history); otherwise, rollback can create inconsistencies between the agent's beliefs and the external world. In RCP-Lite, checkpoints capture workspace artifacts (files and a local SQLite database), policy context (e.g., issued approval tokens), and an agent-provided state snapshot to align runtime control with the formal model $A = (M, \Pi, T, S, E)$.

C. Outcome-grounded micro-suite and balanced scheduling

We constructed a 20-task micro-suite with deterministic graders based on file diffs, structured artifact checks, and SQLite assertions. The suite includes (i) prompt-injection traps targeting tool misuse (e.g., file deletion and destructive SQL), (ii) approval-gated high-risk actions, and (iii) process conformance tests requiring specific tool ordering and mandatory intermediate steps. Episodes are scheduled using a balanced policy to ensure uniform task coverage across runs, reducing variance in aggregate reliability estimates.

D. Experimental validation

We evaluated two agent backends under identical task distributions: (i) a stochastic proxy agent with injected fault modes (repetition, irrelevant actions, missing approvals, and injection-following), and (ii) a hosted LLM agent accessed via API using a constrained JSON action interface. For each backend, we compare baseline execution (no runtime enforcement) against RCP-Lite. Metrics are computed at the episode level and distinguish attempted versus executed violations to separate agent propensity from enforcement effectiveness.

TABLE III STOCHASTIC PROXY RESULTS ON 20-TASK MICRO-SUITE (N=1000, BALANCED SCHEDULE)

Metric	Baseline	RCP-Lite
Task success rate	0.942 (942/1000)	1.000 (1000/1000)
Policy attempt rate	0.086 (86/1000)	0.086 (86/1000)
Executed policy violation rate	0.086 (86/1000)	0.000 (0/1000)
Policy block rate	0.000 (0/1000)	0.086 (86/1000)
Process attempt rate	0.003 (3/1000)	0.003 (3/1000)
Executed process violation rate	0.003 (3/1000)	0.000 (0/1000)
Process block rate	0.000 (0/1000)	0.003 (3/1000)
Budget compliance rate	0.985 (985/1000)	1.000 (1000/1000)
Loop incidence rate	0.071 (71/1000)	0.072 (72/1000)
Rollback incidence rate	0.000 (0/1000)	0.072 (72/1000)
Mean steps to completion	5.062	4.933
Mean tool calls	4.077	3.690
Mean loop detections	0.518	0.137
Mean rollbacks	0.000	0.096

For a hosted LLM backend (DeepSeek) accessed via API, we ran N=50 balanced episodes under conservative decoding settings (temperature=0) and a constrained JSON action interface. Both baseline and RCP-Lite achieved 100% task success with zero attempted or executed violations, indicating that RCP-Lite does not degrade performance for well-behaved agents and adds negligible overhead in this regime. However, this also illustrates a ceiling effect: when attempted violations are rare, enforcement benefits may not be observable at small N; CSE stress variants are therefore recommended to validate control efficacy under elevated stochasticity and adversarial inputs.

TABLE IV HOSTED DEEPSEEK AGENT RESULTS ON 20-TASK MICRO-SUITE (N=50, BALANCED SCHEDULE)

Metric	Baseline	RCP-Lite
Task success rate	1.000 (50/50)	1.000 (50/50)
Policy attempt rate	0.000 (0/50)	0.000 (0/50)
Executed policy violation rate	0.000 (0/50)	0.000 (0/50)
Policy block rate	0.000 (0/50)	0.000 (0/50)
Executed process violation rate	0.000 (0/50)	0.000 (0/50)
Budget compliance rate	1.000 (50/50)	1.000 (50/50)
Mean steps to completion	3.74	3.72
Mean tool calls	2.74	2.72

E. Continuous Sentinel Evaluation (CSE)

CSE operationalizes post-deployment assurance by periodically running the micro-suite (and its adversarial variants) against the live production configuration (model, prompts, tool schemas, and policy rules) without impacting end users. A two-mode design is recommended: (i) a production-realism sentinel that detects regressions under typical settings, and (ii) a stress/adversarial sentinel that increases stochasticity and injects adversarial payloads to validate that the control plane

continues to block unsafe attempts. Sentinel results can drive automated responses such as lowering the allowed autonomy tier, turning off high-risk tools, tightening budgets, or requiring human approval, and failures should be promoted into permanent regression tests via incident-to-eval feedback.

XII. LIMITATIONS AND THREATS TO VALIDITY

First, observability standards for GenAI are evolving; semantic conventions may change as stability levels progress, requiring versioning discipline. Second, benchmark results may not directly transfer to enterprise settings; domain-specific evaluations remain necessary. Third, governance and regulatory obligations vary by jurisdiction and application; the RCP supports evidence generation but does not replace legal interpretation. Finally, trace data can be sensitive; robust redaction and access control are required for safe use.

Ceiling effects can mask enforcement benefits: strong hosted models under conservative decoding and explicit safety prompting may produce near-zero attempted violations, making baseline and RCP performance indistinguishable at small sample sizes. Evaluations should therefore include stress/adversarial variants (e.g., higher stochasticity, prompt variants that do not restate policies, and randomized injection payloads) to validate that the control plane blocks unsafe attempts under non-ideal behavior.

Continuous Sentinel Evaluation bounds behavior under the sentinel task distribution rather than the full production distribution. As with other forms of synthetic monitoring, CSE should be complemented by production telemetry, human review of high-risk actions, and the promotion of incident-to-eval to ensure that the suite evolves in response to real failures and near-misses.

Rollback-based interventions require consistent synchronization of the environment's state and the agent's internal state. While RCP-Lite snapshots both artifacts and agent state, standardizing snapshot/restore interfaces across heterogeneous agent frameworks remains an engineering challenge and may introduce overhead.

Finally, hosted model APIs may change over time (model updates, safety policy shifts, or decoding defaults), which can affect reproducibility. Experiments and sentinel runs should log model identifiers, prompt templates, decoding parameters, tool schemas, and policy versions, and treat these as part of the system configuration under test.

XIII. OPEN RESEARCH DIRECTIONS

1) Trajectory-level verification of policy compliance and safety properties, beyond per-step checks.

2) Standardized session semantics across providers and frameworks for reproducible analysis of agent runs.

3) Governance-aware benchmarks that incorporate permissions, approvals, and data boundary constraints.

4) Standardized security eval suites for tool-using agents aligned to prompt injection and tool abuse risks.

5) Continuous Sentinel Evaluation design: constructing high-signal, low-cost canary suites; calibrating stress variants to observed production failures; and applying statistical process control to detect drift in attempt and intervention rates.

6) Standardized control interfaces for checkpoint/rollback and agent snapshot/restore across heterogeneous runtimes, including consistency guarantees between agent state and external environments.

7) Measurement methods that separate agent propensity from enforcement effectiveness (attempted vs executed violations) and quantify the trade-offs of runtime interventions (false blocks, rollback frequency, and recovery latency).

XIV. CONCLUSION

Agentic AI reliability is fundamentally a systems engineering problem: success depends on long-horizon trajectories, tool interactions, and lifecycle controls rather than single-turn response quality. This paper synthesized evaluation, observability, and governance literature into a unified Reliability Control Plane (RCP) architecture and introduced a reliability metric set that distinguishes attempted from executed violations. A lightweight RCP-Lite implementation and a 20-task outcome-grounded micro-suite illustrate how trace-aware grading and runtime policy enforcement can be operationalized. In a stochastic proxy study (N=1000 balanced episodes), RCP-Lite improved task success and eliminated executed policy, process, and budget violations while preserving the agent's attempted-violation rate, demonstrating enforcement effectiveness independent of agent behavior. Finally, we introduced Continuous Sentinel Evaluation (CSE) as a post-deployment assurance mechanism that continuously re-runs governance-aware canary tests to detect drift and to keep agent autonomy in check over time.

References

[1] OECD, "The Agentic AI Landscape and Its Conceptual Foundations," OECD Artificial Intelligence Papers, 2026. Available: https://www.oecd.org/en/publications/the-agentic-ai-landscape-and-its-conceptual-foundations_396cf758-en.html

[2] M. Mohammadi et al., "Evaluation and Benchmarking of LLM Agents: A Survey," arXiv:2507.21504, 2025. Available: https://arxiv.org/abs/2507.21504

[3] X. Liu et al., "AgentBench: Evaluating LLMs as Agents," in Proc. ICLR, 2024. Available: https://arxiv.org/abs/2308.03688

[4] W. Wang et al., "OdysseyBench: Evaluating LLM Agents on Long-Horizon Complex Office Application Workflows," OpenReview, 2025. Available: https://openreview.net/pdf?id=tMbmBCfSTz

[5] Q. Huang et al., "MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation," in Proc. ICML, 2024. Available: https://cs.stanford.edu/people/jure/pubs/mlagent-icml24.pdf

[6] S. Deng et al., "Mobile-Bench: An Evaluation Benchmark for LLM-based Mobile Agents," in Proc. ACL, 2024. Available: https://aclanthology.org/2024.acl-long.478.pdf

[7] "UltraHorizon: Benchmarking Agent Capabilities in Long-Horizon Partially Observable Environments," OpenReview, 2025. Available: https://openreview.net/pdf/1039b3ebd9b4f107abf59b27e02d59cb3f762 a04.pdf

[8] Anthropic, "Measuring AI Agent Autonomy in Practice," 2026. Available: https://www.anthropic.com/research/measuring-agent-autonomy

[9] Anthropic, "Building Effective Agents," 2024. Available: https://www.anthropic.com/research/building-effective-agents

[10] H. Arksey and L. O'Malley, "Scoping Studies: Towards a Methodological Framework," Int. J. Social Research Methodology, vol. 8, no. 1, pp. 19-32, 2005.

[11] D. Levac, H. Colquhoun, and K. O'Brien, "Scoping Studies: Advancing the Methodology," Implementation Science, vol. 5, no. 69, 2010.

[12] A. C. Tricco et al., "PRISMA Extension for Scoping Reviews (PRISMA-ScR): Checklist and Explanation," Annals of Internal Medicine, vol. 169, no. 7, pp. 467-473, 2018.

[13] Anthropic, "Demystifying Evals for AI Agents," 2026. Available: https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents

[14] OpenAI, "Working with Evals (Evals API)," documentation. Available: https://developers.openai.com/api/docs/guides/evals/

[15] OpenAI, "Evaluation Best Practices," documentation. Available: https://developers.openai.com/api/docs/guides/evaluation-best-practices/

[16] OpenAI, "Testing Agent Skills Systematically with Evals," 2026. Available: https://developers.openai.com/blog/eval-skills/

[17] OpenAI, "OpenAI Evals," software repository. Available: https://github.com/openai/evals

[18] OpenAI, "Tracing - OpenAI Agents SDK," documentation. Available: https://openai.github.io/openai-agents-python/tracing/

[19] OpenTelemetry, "Semantic Conventions for Generative AI Systems," specification. Available: https://opentelemetry.io/docs/specs/semconv/gen-ai/

[20] OpenTelemetry, "GenAI Agent and Framework Spans," specification. Available: https://opentelemetry.io/docs/specs/semconv/gen-ai/gen-ai-agent-spans/

[21] OpenTelemetry, "GenAI Events and Metrics," specification. Available: https://opentelemetry.io/docs/specs/semconv/gen-ai/gen-ai-events/ and https://opentelemetry.io/docs/specs/semconv/gen-ai/gen-ai-metrics/

[22] OpenTelemetry, "Semantic Conventions for OpenAI Client Operations," specification. Available: https://opentelemetry.io/docs/specs/semconv/gen-ai/openai/

[23] OpenTelemetry, "Semantic Conventions for Model Context Protocol (MCP)," specification. Available: https://opentelemetry.io/docs/specs/semconv/gen-ai/mcp/

[24] NIST, Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1, 2023. Available: https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf

[25] NIST, AI RMF: Generative Artificial Intelligence Profile, NIST AI 600-1, 2024. Available: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

[26] IMDA (Singapore), Model AI Governance Framework for Agentic AI, Version 1.0, 2026. Available: https://www.imda.gov.sg/-/media/imda/files/about/emerging-tech-and-research/artificial-intelligence/mgf-for-agentic-ai.pdf

[27] ISO/IEC, ISO/IEC 42001:2023 - Artificial Intelligence Management System, 2023. Available: https://www.iso.org/standard/81230.html

[28] European Union, Regulation (EU) 2024/1689 (Artificial Intelligence Act), Official Journal of the European Union, 2024. Available: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

[29] OWASP, "Top 10 for Large Language Model Applications," 2025. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/

[30] OWASP GenAI Security Project, "LLM01: Prompt Injection," 2025. Available: https://genai.owasp.org/llmrisk/llm01-prompt-injection/