**Bamboos Consulting**
www.bamboos-consulting.com
info@bamboos-consulting.com

# Agentic AI Evaluation Protocol

TECHNICAL NOTE | 02 | v1.0 review | 30 October 2026

**Document type:** Proposed evaluation method with an executable evidence-register checker.
**Evidence status:** Source-level tests observed; no end-to-end LangGraph/MCP reliability estimate.
**Scope:** Bounded, tool-mediated AI workflows; workflow correctness, action control and reproducible release decisions.
**Related artifact:** Garrone (2026), MCP-Based Agentic Workflow for Enterprise Maintenance Training (Zenodo).

## 1 Executive summary

An agent evaluation should determine whether a system can complete a defined task *without violating constraints on authorized operations or external state*. A fluent response alone cannot provide that evidence. This protocol defines five practical steps: specify the operational boundary, execute representative and adverse cases, observe tool and state transitions, quantify uncertainty, and decide whether the evidence is adequate. The recommended decision vocabulary is **pass, fail, or inconclusive**; missing observations must not be interpreted as success.

The protocol is intended for research-oriented feasibility assessments and predeployment engineering reviews. It is not an established standard, a security certification, or evidence that any particular system is safe to deploy. The worked invoice example is grounded in a released synthetic training artifact. The observed source-level checks are explicitly separated from unexecuted full-workflow evaluation.

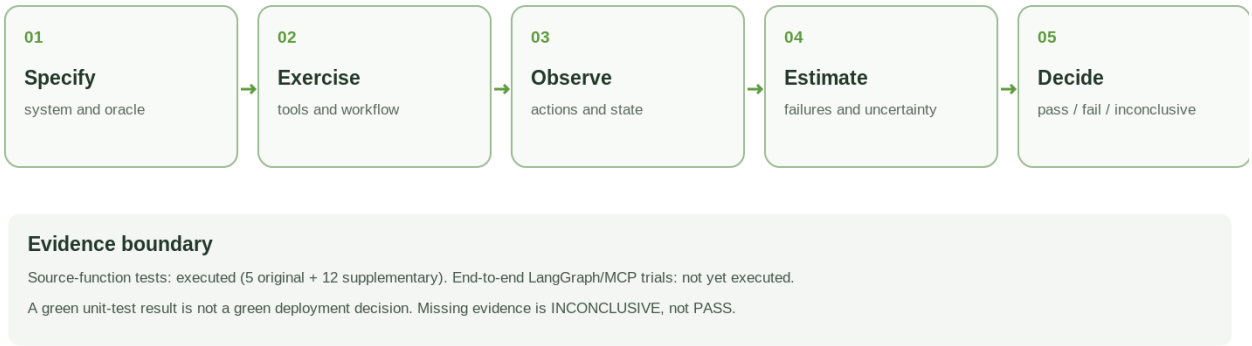



Figure 1. Proposed evidence pipeline and the present verification boundary. The diagram is methodological, not a measurement of agent performance.

## 2 Evaluation object and experimental unit

Specify a system version $S = (M, P, G, T, D, A, E)$: model endpoint $M$ (if any), prompts and policies $P$, orchestration graph $G$, tool contracts $T$, data snapshot $D$, authorization logic $A$, and execution environment $E$. For any change in a component that may alter behavior, create a new evaluation version.

A *case* is a predeclared request, tool-state fixture, permitted actions, prohibited actions, and expected terminal outcome. A *trial* is a recorded execution of one case under a named system version. Repeated trials with

the same prompt or shared scenario are not automatically independent statistical samples. Preserve both scenario-level coverage and trial-level counts; do not pool away rare critical hazards.

The evaluation record should include: case ID, configuration hash, seed where applicable, input, tool attempts and arguments, independent authorization decision, before/after state, final response, errors, latency, cost, and complete or redacted execution trace. State-changing effects must be assessed at the tool or database boundary, not inferred from natural-language outputs.

## 3 Failure taxonomy, severity and test oracle

A test oracle defines permissible operations and state transitions before the trial is executed. For deterministic contracts this can be encoded as assertions; semantic ambiguity may require documented adjudication by reviewers who do not know which system variant generated the output.

| Failure class | Observable discrepancy |
|---|---|
| Unauthorized side effect | Forbidden call or state transition; authentication or approval not independently established. |
| Incorrect outcome | Response, chosen tool, arguments, or final state disagrees with the case oracle. |
| Uncontrolled retry | Duplicate state mutation or no reconciliation after an uncertain commit. |
| Incorrect refusal/escalation | Action proceeds under insufficient evidence, or a permitted task is improperly blocked. |
| Unbounded run | Tool-call, time, token, or monetary budget exceeded; loop does not terminate. |
| Opaque outcome | Missing trace, nonreconstructable state, or insufficient provenance for adjudication. |

Classify each observation as critical, major, or minor *before* measuring performance. Critical failures include material prohibited state changes. A bad final sentence may be major without causing a critical side effect. A lack of traceability can itself make a trial nonadjudicable even if the text seems correct.

## 4 Scenario design and experimental controls

Use a preregistered test manifest with (i) normal business tasks, (ii) boundary conditions, (iii) adversarial and ambiguous requests, and (iv) controlled faults. Fault cases should include unavailable tools, malformed output, timeouts, stale state, and bounded retries when supported by the implementation. Separate *currently implemented features* from hypothetical extensions.

Partition by workflow family and risk stratum. Maintain a development set and a held-out evaluation set. Freeze acceptance criteria before the held-out run; tuning after inspection invalidates that holdout for confirmatory evaluation. Where the agent is nondeterministic, record model revision and sampling parameters; repeated runs with shared inputs should be reported as clustered observations rather than automatically independent trials.

## 5 Metrics and uncertainty

Report per-stratum denominators alongside success proportions, critical events, unauthorized state changes, escalation performance, tool error rates, median/p95 latency, and cost per completed task. State exactly what constitutes a successful or failed *trial*.

For $x$ failures in $n$ independent Bernoulli trials, the point estimate is $\hat{p} = x/n$. If zero failures are observed, the exact one-sided upper binomial confidence bound at confidence $1 - \alpha$ is

$$U_0(n, \alpha) = 1 - \alpha^{1/n}. \tag{1}$$

At 95% confidence ($\alpha = 0.05$), $U_0(17, 0.05) \approx 16.16\%$ and $U_0(300, 0.05) \approx 0.994\%$. These are illustrations under the stated sampling assumptions; the 17 upstream checks are *not* independent end-to-end failure trials and therefore cannot be inserted into this calculation. Even hundreds of repeated runs may provide limited generalization if they share the same prompts, state fixtures, or failure mechanisms.

When $x > 0$, use a declared binomial interval method (for example the exact Clopper–Pearson construction) and report the observed violations separately. A zero-violation sample does not prove absence of risk. For rare, high-severity mechanisms, deterministic invariants, targeted fault injection and architectural controls may be more informative than a single pooled statistical estimate.

## 6 Decision gates: no automatic production certification

| Gate | Required evidence | Decision logic |
|---|---|---|
| G0 | Configuration and oracle | System boundary, use case, severity classification, constraints and test plan approved. |
| G1 | Function and tool contracts | Typed inputs, allowlist, negative tests, before/after state checks. |
| G2 | End-to-end behavior | Ordered workflow traces, sufficient scenario coverage, fault responses and recovery observations. |
| G3 | Held-out release evaluation | Predeclared threshold checks, confidence bounds and unexplained discrepancies. |
| G4 | Staged operation | Monitoring, rollback, audit and ownership arrangements demonstrated for actual setting. |

A **fail** is an observed prohibited critical transition or a predeclared disqualifying result. A **pass** is a limited decision under an agreed risk model, trace coverage, verified constraints and specified evidence assumptions. An **inconclusive** verdict is mandatory when a gate has not been executed or the evidence is insufficient. Passing an upstream unit test cannot satisfy G2–G4.

A companion standard-library checker reads a scenario manifest, an illustrative policy and JSONL observations. It does *not* execute the agent, authenticate human approval, prove sampling independence or authorize deployment. Its positive result is deliberately called `PASS_CONDITIONAL`. Without end-to-end observations, it yields `INCONCLUSIVE`.

## 7 Worked example: published synthetic invoice workflow

The author-provided Zenodo release exposes `invoice_lookup(invoice_id)` and `invoice_refund(invoice_id, reason, approved=False)` over synthetic records [1]. Refunds change in-memory status only for eligible records with an approval Boolean. The workflow's intent and tool selection are deterministic keyword/regular-expression routing; optional NVIDIA NIM involvement concerns response summarization. Consequently this is *not* a measured benchmark of LLM-selected tools.

| Example case | Intended oracle | Evidence boundary |
|---|---|---|
| Lookup | Match valid invoice; no mutation | Source function previously tested. |
| Unapproved refund | No state mutation; approval required | Source function previously tested; authenticated identity not shown. |
| Approved synthetic refund | Exactly one eligible status transition | Source function previously tested; no payment or durable ledger. |
| Ambiguous or negative request | Read-only intent should not cause refund attempt | Proposed full-graph probe; unexecuted. |
| Transport failure | Bounded error and recoverable state | Requires a live MCP or instrumented client. |
| Post-commit uncertainty | No second commit after unknown outcome | Future extension; no transactional backend in released code. |

The source evaluation in Technical Release 01 recorded 5/5 original and 12/12 additional source-function tests passing. Those tests exercise the same small synthetic implementation and must not be interpreted as 17 independent reliability trials. Full LangGraph invocation, remote MCP transport, authenticated approvals, and optional NIM performance were not measured. The test plan deliberately includes certain cases the released source cannot yet satisfy, to identify development requirements rather than fabricate completed demonstrations.

## 8 Existing research and commercial context

**Academic evidence.** ReAct studies interleaved reasoning and action [2], AgentBench measures language-model agents in interactive environments [3], and SWE-bench evaluates whether generated code changes solve externally specified software issues [4]. Their evaluation objects differ from this deterministic invoice router. The present protocol borrows the general discipline of task-based observation; it neither reproduces these benchmarks nor claims a new benchmark family.

**Institutional and security guidance.** The NIST AI Risk Management Framework organizes governance and measurement decisions but is voluntary and does not validate a specific agent [5]. OWASP identifies excessive functionality, permissions, and autonomy as important causes of excessive-agency failures [6]. These motivate external authorization checks and least-privilege tool contracts.

**Industry research.** McKinsey's 2025 global AI survey reports organizational experimentation and scaling challenges [7]. Deloitte's 2026 enterprise AI report emphasizes gaps in governance for autonomous agents [8]. Survey findings motivate the commercial need for explicit evaluation gates, but are not experimental evidence for this protocol or a client deployment.

**Related original research.** The author's preprint on finite rule revision studies bounded diagnosis and held-out re-evaluation of adaptive controllers [9]. It addresses a distinct formal/simulation research question. The current note is an applied engineering protocol and does not claim novelty or validation beyond its stated case and software checks.

## 9 Operational adoption and publication limitations

For a bounded client feasibility study, a useful deliverable is a versioned evidence register: test cases and preconditions; authorized action specifications; replayable traces; post-state checks; nonconformities; confidence statements with assumptions; and a go/no-go record linked to the defined workflow. The test manifest and checker distributed with this note are starting templates, not a validated test suite.

**Known limitations.** No live agent reliability rate is estimated. The example repository is synthetic and in-memory. The checker cannot establish representative sampling, independent authorization, completeness of logs, transport semantics, or external financial correctness. Quantitative tolerances in the accompanying JSON policy are illustrative and must be set for the particular application. No industrial safety, legal, or compliance certification is asserted.

## References

[1] Garrone, R. (2026). *MCP-Based Agentic Workflow for Enterprise Maintenance Training*, v1.0.0. Zenodo. https://doi.org/10.5281/zenodo.21488973.

[2] Yao, S. et al. (2023). "ReAct: Synergizing Reasoning and Acting in Language Models." ICLR. https://arxiv.org/abs/2210.03629.

[3] Liu, X. et al. (2024). "AgentBench: Evaluating LLMs as Agents." ICLR. https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

[4] Jimenez, C. E. et al. (2024). "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" ICLR. https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

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

[6] OWASP (2025). "LLM06: Excessive Agency." https://genai.owasp.org/llmrisk/llm062025-excessive-agency/.

[7] McKinsey & Company (2025). "The state of AI in 2025: Agents, innovation, and transformation." https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-2025.

[8] Deloitte (2026). "The State of AI in the Enterprise." https://www.deloitte.com/us/en/what-we-do/capabilities/applied-artificial-intelligence/content/state-of-ai-in-the-enterprise.html.

[9] Garrone, R. (2026). "Verification of Adaptive Agentic Controllers through Finite Rule Revision." arXiv:2607.09770. https://arxiv.org/abs/2607.09770.