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

Evaluating an MCP-Based Invoice Workflow

TECHNICAL NOTE | Technical Case Study 01 | v1.0 | 30 October 2026

Document type: Reproducible source-level assessment and methodological case study. **Evidence status:** Verified tool-function behavior in a synthetic repository; complete LangGraph execution and live MCP transport not measured.
Primary artifact: Garrone (2026), *MCP-Based Agentic Workflow for Enterprise Maintenance Training*, Zenodo v1.0.0, DOI:10.5281/zenodo.21488973.

1 Executive summary

A plausible language-model response is not sufficient evidence that an agent has performed an authorized business operation. Assessment should distinguish request interpretation, orchestration, tool invocation, authorization checks, state transition, and output generation. This case study examines an independently executable subset of a publicly archived training artifact that implements synthetic invoice lookup and refund operations. The published code uses a deterministic request router: model-based reasoning is *not* responsible for selecting the financial tool. NVIDIA NIM is an optional response-summarization component.

Five original invoice-tool tests and twelve additional source-function checks passed in the v0.7 evaluation environment. These are checks of a small, fixed, synthetic system: neither 17 statistically independent cases nor an estimate of production reliability. A prepared end-to-end LangGraph runner has not been executed, and no MCP network-service or NIM performance results are reported.

2 Business question and evaluation boundary

A service organization considering automated record handling needs to know whether a system selects an appropriate operation, supplies valid arguments, rejects unauthorized changes, and maintains consistent business state. These questions are separable. A correct final sentence does not establish that a refund was never committed; a rejected state transition does not establish that every interpretation of a user request is correct.

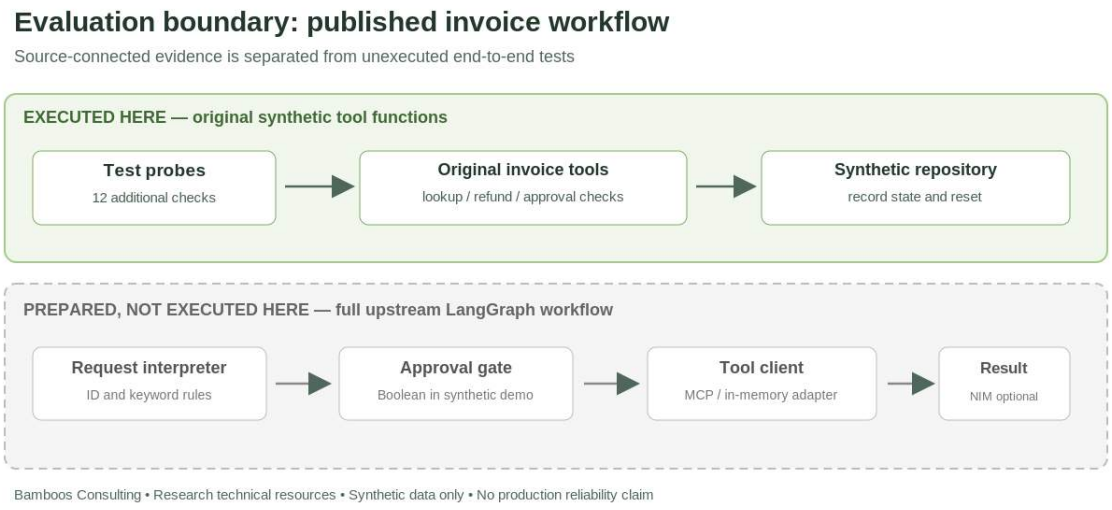


Figure 1. Source-function checks are completed; LangGraph and live MCP evaluations remain separate unexecuted stages.

3 Examined implementation and reproducibility

The source archive supplied by the author contains `src/agent/workflow.py`, `src/agent/nodes.py`, an in-memory tool adapter, MCP client and server code, and a synthetic invoice repository. The workflow builds a LangGraph state graph with interpretation, approval, tool execution, and response nodes. In the interpreter, a regular expression extracts a record identifier and a keyword list chooses `invoice_lookup` or `invoice_refund`. The possible false-positive effects of keyword routing warrant direct evaluation; they have not been quantified by the recorded tool-function tests.

The independent checks import `src.mcp_server.tools` without replacing the functions, initialize the synthetic records, invoke public functions, and inspect returned statuses and before/after state. The evaluation reports the SHA-256 checksum of the upstream `tools.py` file so the source can be identified. Each state-dependent check resets records to avoid cross-case leakage. The original archive should remain a separate, cited research artifact; this evaluation does not rebrand the upstream code as a new agent implementation.

4 Observed test results

Test tier	Result	Inference supported
Upstream invoice-tool tests	5/5 passed	Original source-level unit-test expectations met.
Additional function probes	12/12 passed	Status, validation, approval-flag and synthetic-state properties in the fixed scenarios.
Evaluation-harness unit tests	4/4 passed	Harness scoring logic; not workflow performance.
Native LangGraph invocation	Not run	No observed end-to-end tool selection or graph execution results.
MCP transport and optional NIM	Not run	No network, latency, external endpoint or model claims.

The twelve additional probes addressed case-insensitive lookup, unknown records, missing approval, approved refund, duplicate refund, ineligible record, previously refunded record, absent invoice, malformed identifier, nonallowlisted tool, non-Boolean approval, and repository reset. In these tests an unapproved refund did not change the record; an approved eligible refund changed the record to `refunded`; a repeat returned `already_refunded`. These observations support correctness of those specific functions under those inputs, not exhaustive security or transactional guarantees.

5 Crucial limitations in the implementation

Approval is not identity verification. The refund function takes `approved: bool`. A true value supplied by the caller is not proof of independently authenticated human consent. An operational system requires an independently verified principal, provenance of approval, a bounded authorization scope, and auditable enforcement outside the language-model prompt. Relevant design risks include excessive tool authority and insecure downstream actions [5].

The interface is narrower than generic refund workflows. The source defines `invoice_refund(invoice_id, reason, approved=False)`. It does not accept a refund amount, enforce a monetary ceiling, implement a durable ledger, or resolve a timeout after a possibly committed transaction. Earlier illustrative scenarios concerning amount ceilings or post-commit recovery are proposals for extension, not characteristics of the released source.

Request classification is deterministic. Keyword matching differs fundamentally from an LLM deciding which tool to invoke. In particular, a read-only request containing the word `refund` could be routed as an action request. A prepared adversarial probe addresses this possibility, but its full-graph result remains unmeasured. Accordingly, the study is not an LLM planning benchmark.

6 Evidence-oriented evaluation sequence

A staged assessment should proceed from source verification to external-effect observation. In Stage A, record exact source, test fixtures, and tool schemas. In Stage B, repeat tool-function tests, including negative cases

and synthetic state reset. In Stage C, execute the original LangGraph application offline and record ordered tool calls, normalized inputs, final state, errors, and state transitions. In Stage D, repeat through a live MCP server under controlled transport faults. In Stage E, add independent authentication, audit trails, concurrency, and commit/retry tests in a separate extension branch.

Each stage requires a predeclared oracle. For example, an unauthorized refund request should produce no refund state transition; an eligible approved synthetic refund should produce one transition; a repeated refund should not produce a second transition. A stage should be labeled *not run* rather than *passed* when no trace has been collected.

7 Relationship to research and industry evidence

AgentBench evaluates language-model agents in interactive environments, making it relevant to the general principle of observing action outcomes [2]. ReAct links language reasoning to externally visible action trajectories [3]; neither work supplies validity evidence for this deterministic router. The NIST AI Risk Management Framework is a voluntary governance framework and can help organize evidence requirements; it is not a certification of a particular implementation [4]. OWASP's excessive-agency guidance contextualizes tool authorization limits [5].

McKinsey's 2025 survey describes organizational experimentation and difficulties scaling AI, providing context for why staged evaluation is commercially relevant [6]. Survey responses do not measure this workflow's reliability or indicate that the present case represents deployed enterprise software. The author's distinct work on adaptive-controller rule revision addresses a different research question and should not be conflated with the source-function results here [7].

8 Implications for industrial feasibility studies

The transferable deliverable is an evidence register: versioned test cases, explicit expected state transitions, executed traces, boundary-specific pass/fail records, and a release decision tied to known limitations. For SMEs, this may be more useful than an undifferentiated agent accuracy percentage. Subsequent feasibility studies can separately evaluate tool authorization, data integrity, transport robustness, recovery semantics, and deployment costs on a defined infrastructure.

Disposition. This case study is suitable for public release as a *source-level methodological demonstration*, once links and the paired website article are checked. It is not evidence of production readiness, complete LangGraph functionality, validated MCP network operation, or performance of an autonomous financial agent.

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

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

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

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

[6] 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.

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