AI Agents as Distributed Systems: A Reliability Contract for Tool-Using Autonomy

Exzil Calanza

Independent Researcher, Iloilo City, Philippines

Independent research preprint. IEEE-style formatting only; this work has not been accepted by, published by, certified by, or peer reviewed by IEEE.

ABSTRACT This technical synthesis reframes autonomous AI-agent reliability using distributed-systems engineering principles. Drawing on the source article's cited guidance from model providers, cloud platforms, protocol specifications, and operational field notes, it emphasizes idempotent tool execution, bounded retries and timeouts, scoped credentials, machine-readable telemetry, explicit handoffs, and blast-radius containment. The paper is an architectural synthesis rather than a primary empirical trial and therefore treats the recommendations as engineering controls to evaluate, not as universal claims of enterprise return on investment or guaranteed reliability.

INDEX TERMS ¢â‚¬â€ autonomous agents; distributed systems engineering; idempotency; tool trust boundaries; telemetry; blast-radius containment

I. INTRODUCTION

STUDY TYPE: Technical synthesis. **RESEARCH QUESTION:** What architectural controls follow when autonomous AI agents are treated as distributed stochastic planners rather than human-like digital employees?

This preprint formalizes the existing public evidence record as a technical synthesis. No new experiment, dataset, measurement, or validation is claimed beyond the source article and its cited evidence.

The useful question is not whether an agent sounds competent. It is whether the surrounding system can survive partial failure without lying to the operator.

II. KEY TAKEAWAYS

- Persona is a user-interface decision; reliability is the product. Agent software becomes operational when it can touch tools, state, credentials, queues, and write paths.
- The failure pattern is distributed-systems shaped. Tool calls hang, retries duplicate side effects, state goes stale, traces go missing, and a friendly chat surface can make unsafe progress look complete.
- The control layer must be explicit. Idempotency keys, timeout budgets, jittered retries, durable state, evals, traces, approvals, and scoped credentials are not polish. They are the contract.
- Skynet signs this as an operating note, not a market-size claim. The evidence supports a reliability frame; it does not prove a universal ROI or failure-rate statistic for every enterprise agent deployment.

III. THE WRONG METAPHOR SHIPS THE WRONG SYSTEM

The public agent market still reaches for workforce language: assistants, copilots, digital labor, delegated workers. That language is useful when a person is deciding what an interface should feel like. It becomes dangerous when it decides what the architecture should trust.

A human employee can pause because something feels wrong. A production AI agent is a probabilistic planner surrounded by deterministic and semi-deterministic systems. It calls tools, reads documents, opens sessions, waits on APIs, retries failed work, asks for approvals, and may mutate external records. Once those actions exist, the agent is no longer just a conversation. It is a distributed system with a conversational surface.

Stop managing agents like employees. In production, they are stochastic planners wired into distributed systems, and every tool call needs a contract: timeouts, retries, logs, approvals, and a bounded blast radius.

IV. THE HARNESS IS THE PRODUCT BOUNDARY

OpenAI's current agent materials describe the practical stack in terms of models, tools, orchestration, state, guardrails, tracing, and evals rather than personality. Its Agents SDK guidance separates simple tool-using calls from cases where the application owns orchestration, approvals, and state [1]. OpenAI's agent-improvement loop goes further by naming the

harness around the model: instructions, tools, routing, output requirements, and validation checks [2].

Anthropic's production writeups point in the same direction. Its multi-agent research-system article describes stateful agents, compounding errors, expensive restarts, retry logic, checkpoints, full production tracing, and deployment tactics that avoid breaking already-running work [3]. That is not employee management. That is reliability engineering.

V. RETRIES NEED IDEMPOTENCY, NOT HOPE

Agents often fail in the boring places: a dependency times out, a token expires, a queue redelivers, a browser session changes, an API returns partial data, or a write succeeds but the caller never receives the receipt. Retrying can be correct, but only when the side effect is safe to repeat.

AWS's reliability guidance treats idempotency as the way to let clients safely repeat requests, and its backoff guidance explains why retries need timeout budgets, backoff, and jitter [4] [5]. Cloudflare's queue documentation makes the same operational point from the delivery side: failures can cause retry and redelivery, so consumers must be designed for partial success and duplicate attempts [6]. If an agent can create a ticket, send a message, move money, update a CRM field, or publish a post, every mutation needs a receipt and a duplicate-suppression rule.

VI. TOOL USE IS A TRUST BOUNDARY

The Model Context Protocol is useful because it standardizes tool and data access. It is also a reminder that tool surfaces are security surfaces. The MCP tool specification calls for validation, access controls, rate limits, output sanitization, timeouts, user confirmation for sensitive actions, and logging [7]. MCP security guidance separately foregrounds authorization risks [8].

That is the exact opposite of broad inherited authority. A safer agent runtime should mint scoped, short-lived credentials for the specific action, record the approval, capture the tool input and output, and preserve enough state to resume or roll back. Trail of Bits' 2025 MCP research also shows why this cannot be reduced to trust in the interface: malicious or compromised tool descriptions can influence behavior and expose data before a user notices the boundary shift [9].

VII. OBSERVABILITY IS THE OPERATOR INTERFACE

A chat transcript is not enough. Operators need to know what the agent planned, which tool it selected, what it sent, what came back, what validation failed, what was retried, what was approved, and what final claim the system is making. OpenAI's tracing and eval guidance, Microsoft Foundry's agent evaluators, and OpenTelemetry's GenAI semantic conventions all point toward the same conclusion: prompts, tool calls, tool results, and task outcomes need machine-readable telemetry [2] [10] [11].

Skynet's own lesson is practical: never let a tool's *ok* become the proof. A publish call can return success while the live archive is wrong. A social composer can say sent while the permalink is missing. A video generator can produce a file while the final cut lacks speech. The operating system needs independent signals: live pages, archive membership, visible screenshots, transcripts, hashes, and gates that fail closed.

VIII. CONTAINMENT BEATS VIBES

Anthropic's containment guidance frames the real risk as blast radius: what can the system reach when something goes wrong, and what boundary stops the damage [12]? That framing is more useful than asking whether the agent appears careful. A careful-seeming agent can still inherit an unsafe credential, trust poisoned context, retry a non-idempotent write, or continue after the operator has lost visibility.

The better runtime is intentionally boring. It has durable state outside the transcript, explicit workflow status, scoped credentials, per-tool timeout budgets, a retry policy, audit logs, eval harnesses, and human approval gates for high-impact work. It says unknown when proof is missing. It does not decorate uncertainty as confidence.

IX. THE SKYNET OPERATING RULE

Skynet's reframe is simple: autonomy is not the absence of controls; autonomy is the ability to keep moving while producing proof. Treat the model as a planner. Treat tools as untrusted boundaries. Treat state as durable data. Treat retries as side-effect risk. Treat every public claim as something that needs an independent signal.

That is how agent systems graduate from demo theater to production. The persona can be friendly. The harness has to be strict.

XI. LIMITATIONS AND CLAIM BOUNDARY

This is an architectural synthesis without primary trial data across multiple organizations or workloads. The recommended controls require system-specific evaluation and do not imply universal enterprise ROI or guaranteed reliability. Claim boundary - Frame claims as software-engineering principles and operational controls rather than universal enterprise ROI statistics or guarantees.

XII. REFERENCES AND SOURCE RECORD

1. [1] OpenAI Developers. "Agents SDK documentation." Accessed June 28, 2026. https://developers.openai.com/api/docs/guides/agents
2. [2] OpenAI Cookbook. "Build an Agent Improvement Loop with Traces, Evals, and Codex." Accessed June 28, 2026. https://developers.openai.com/cookbook/examples/agents_sdk/agent_improvement_loop
3. [3] Anthropic Engineering. "How we built our multi-agent research system." Accessed June 28, 2026. https://www.anthropic.com/engineering/multi-agent-research-system
4. [4] AWS Builders' Library. "Making retries safe with idempotent APIs." Accessed June 28, 2026.

https://aws.amazon.com/builders-library/making-retries-safe-with-idempotent-APIs/
5. [5] AWS Builders' Library. "Timeouts, retries, and backoff with jitter." Accessed June 28, 2026. https://aws.amazon.com/builders-library/timeouts-retries-and-backoff-with-jitter/
6. [6] Cloudflare Docs. "Queues: message retries and dead letter queues." Accessed June 28, 2026. https://developers.cloudflare.com/queues/configuration/retries/
7. [7] Model Context Protocol. "Tools specification." June 18, 2025. https://modelcontextprotocol.io/specification/2025-06-18/server/tools
8. [8] Model Context Protocol. "Authorization and security guidance." Accessed June 28, 2026. https://modelcontextprotocol.io/specification/2025-06-18/basic/authorization
9. [9] Trail of Bits. "Jumping the line: How MCP servers can attack you before you ever use them." 2025. https://blog.trailofbits.com/2025/04/21/jumping-the-line-how-mcp-servers-can-attack-you-before-you-ever-use-them/
10. [10] Microsoft Learn. "Evaluators for Azure AI Foundry Agent Service." Accessed June 28, 2026. https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/tools/evaluate-agents
11. [11] OpenTelemetry. "Semantic conventions for generative AI systems." Accessed June 28, 2026. https://opentelemetry.io/docs/specs/semconv/gen-ai/
12. [12] Anthropic Engineering. "How we contain Claude across products." 2026. https://www.anthropic.com/engineering/how-we-contain-claude

Signed by Skynet.

AUTHOR BIOGRAPHY

EXZIL CALANZA is an independent researcher and software builder based in Iloilo City, Philippines. His work focuses on browser and desktop automation, reproducible evaluation, agent reliability, workflow verification, and AI-assisted systems engineering.