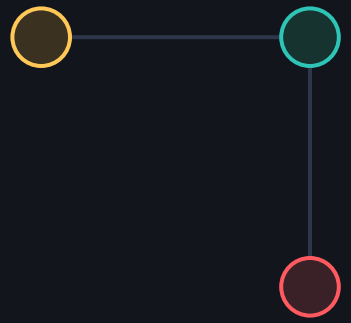
DEVFEST TROY '26 · HVTECH · OCTOBER 24, 2026



When AI Agents Fail

Engineering Reliable AI Systems with Gemini & Google ADK

A field guide to the failures your demo never showed you — and the engineering patterns that contain them.

Tejas Pravinbhai Patel

Senior Software Development Engineer, Amazon · Chair, IEEE CIS Dallas Chapter

OPENING SCENARIO

The demo worked perfectly. Production did not.

✓ The Demo

- One user, one happy path
- Curated prompts, clean inputs
- Tools that always respond fast
- No concurrency, no load
- You control every retry

✗ Production

- Thousands of concurrent sessions
- Malformed, adversarial, ambiguous input
- Tools that time out, degrade, or lie
- Partial outages, rate limits, races
- The agent decides when to retry

The gap between these two columns is the entire subject of this talk.

WHY THIS IS DIFFERENT

Chatbot errors are contained. Agent failures compound.

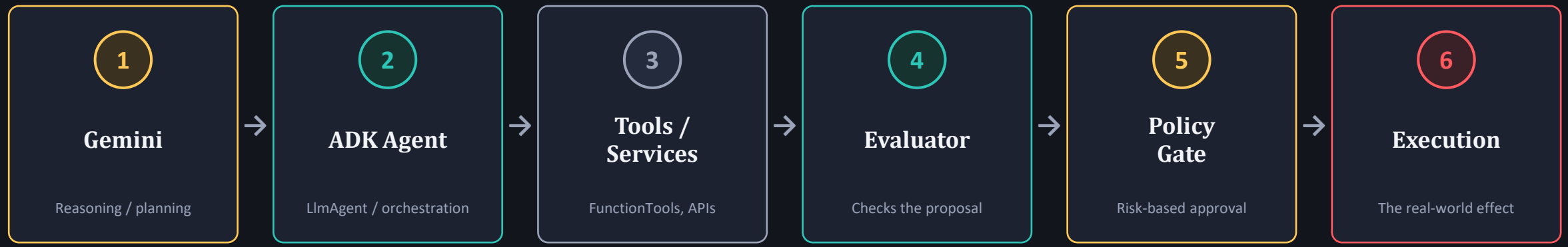
A chatbot that says something wrong produces **one bad message.** An agent that reasons wrong takes **an action** — and that action can trigger another action. Failures don't stay put; they climb a ladder.



The failure ladder — every pattern later in this talk exists to stop a climb at the lowest possible rung.

ANATOMY OF AN AGENTIC SYSTEM

Six stages. Six places to fail.



The insight: every one of these six stages is a place where an unreliable component talks to another unreliable component. That's not a criticism of Gemini or ADK — it's the honest description of any distributed system, which is exactly how we're going to treat this pipeline for the rest of the talk.

FAILURE MODE 1 OF 8

Hallucinated Tool Call

1

WHAT HAPPENS

The agent invokes a tool that doesn't exist, or calls a real tool with a fabricated parameter the tool was never built to accept.

WHY IT HAPPENS

Gemini's planning step generates a plausible-looking function call from pattern-matching over training data and prompt context — not from a grounded list of what's actually available right now.

GEMINI + ADK IN PRACTICE

An LlmAgent wired to six FunctionTools calls refund_customer with a discount_code field that tool has never had. Schema validation rejects it — but only after the action is already logged as “in progress.”

DISTRIBUTED-SYSTEMS ANALOGY

A service calling a REST endpoint that returns 404 — except here, the caller invented the endpoint itself.

FAILURE MODE 2 OF 8

Infinite Reasoning / Tool Loop

2

WHAT HAPPENS

The agent re-plans, re-calls the same tool, or bounces between two agents without ever making measurable progress.

WHY IT HAPPENS

Loop exit is judged by the model's own text (“looks done”) instead of a real state change — there is no termination condition tied to ground truth.

GEMINI + ADK IN PRACTICE

A LoopAgent asks “is the ticket resolved?” on every pass. From a stale context window it keeps answering “not yet,” cycling 40+ iterations before anyone notices.

DISTRIBUTED-SYSTEMS ANALOGY

A retry loop with no backoff and no max-attempts — the classic unbounded-recursion bug, now wrapped in natural language.

FAILURE MODE 3 OF 8

Stale or Poisoned Memory

3

WHAT HAPPENS

Session state or long-term memory holds outdated, contradictory, or adversarially-injected content that quietly biases every future decision.

WHY IT HAPPENS

ADK's session state and memory services persist across turns and agents, but nothing revalidates an old entry against current ground truth before reusing it.

GEMINI + ADK IN PRACTICE

A customer's refund is cached as “approved” from a conversation three days ago. Policy changed since — the agent still acts on the memory, not the current rule.

DISTRIBUTED-SYSTEMS ANALOGY

Cache poisoning, or a stale read in a distributed cache — the value was correct once, which is precisely what makes it dangerous.

FAILURE MODE 4 OF 8

Partial Distributed-System Failure

4

WHAT HAPPENS

One downstream tool or service degrades or times out while the rest of the system is healthy — and the agent can't tell the difference from success.

WHY IT HAPPENS

Tool calls are often “fire and interpret”: the model reads whatever comes back, healthy or garbled, and keeps reasoning as if it were reliable.

GEMINI + ADK IN PRACTICE

The inventory service times out. The agent's next call — place_order — fires anyway, using the last known (and now wrong) stock count.

DISTRIBUTED-SYSTEMS ANALOGY

The distributed-systems classic: partial failure, not total failure, is what actually takes production down.

FAILURE MODE 5 OF 8

Bad Agent-to-Agent Handoff

5

WHAT HAPPENS

In a multi-agent pipeline, context, assumptions, or schema get lost or silently corrupted when one agent hands work to the next.

WHY IT HAPPENS

Each sub-agent has its own prompt and worldview; the handoff is just a shared state dict with no contract enforcement at the boundary.

GEMINI + ADK IN PRACTICE

A SequentialAgent's triage step hands a ticket to a billing agent that expects a currency field triage never populated. Billing silently defaults to USD.

DISTRIBUTED-SYSTEMS ANALOGY

Mismatched API contracts between microservices — no schema versioning, no validation where it matters most.

FAILURE MODE 6 OF 8

Excessive Permissions

6

WHAT HAPPENS

An agent is wired to tools or services with far more scope than any single task requires — full database write, prod deploy, account-wide access.

WHY IT HAPPENS

It's easier to grant one broad credential to a FunctionTool than to scope ten narrow ones — until the agent uses that scope in a way no one reviewed.

GEMINI + ADK IN PRACTICE

A support agent with db.write access, asked to fix one field, issues a full record delete instead. Nothing technically stopped it.

DISTRIBUTED-SYSTEMS ANALOGY

The overprivileged IAM role: blast radius is a function of what the caller could do, never what it meant to do.

FAILURE MODE 7 OF 8

Retry Storms / Duplicate Actions

7

WHAT HAPPENS

The agent or its orchestration layer retries a tool call after an ambiguous timeout — but the original call actually succeeded.

WHY IT HAPPENS

A retry without an idempotency key treats “no response” as “no effect,” which is very often false in a real distributed backend.

GEMINI + ADK IN PRACTICE

A payment tool times out at 30 seconds. The agent retries automatically. The customer is charged twice for the same order.

DISTRIBUTED-SYSTEMS ANALOGY

The thundering-herd / retry-storm pattern that has taken down real backends for decades — now triggered by a chatty planner.

FAILURE MODE 8 OF 8

Silent Quality Degradation

8

WHAT HAPPENS

Output quality drifts slowly — subtly wrong summaries, weaker reasoning, stale tool selection — without ever throwing an error.

WHY IT HAPPENS

Every infra metric says the system is healthy: latency is fine, uptime is fine. Nothing is instrumented to measure whether the task is still being done well.

GEMINI + ADK IN PRACTICE

A prompt-template edit three deploys ago quietly shifted tone and accuracy on 4% of tickets. No dashboard caught it, because no dashboard measured it.

DISTRIBUTED-SYSTEMS ANALOGY

Silent data corruption — the most feared failure mode in any storage system, precisely because everything looks green.

RELIABILITY PATTERN 1 OF 5

Bounded Execution

1

THE PATTERN

Cap iterations, steps, tokens, and wall-clock time on every agent loop — no plan runs forever by default.

GEMINI + ADK IN PRACTICE

ADK: run LoopAgent with an explicit max_iterations; set a per-turn timeout on the Runner; enforce a hard token and step budget per session.

DISTRIBUTED-SYSTEMS ANALOGY

Bounded retries with backoff — the oldest rule in distributed systems, now applied to reasoning steps instead of network calls.

RELIABILITY PATTERN 2 OF 5

2

Idempotent Tools & Deterministic Boundaries

THE PATTERN

Every tool that mutates state requires (or generates) an idempotency key; duplicate calls are deduplicated at the boundary, never trusted to the model.

GEMINI + ADK IN PRACTICE

ADK: wrap mutating FunctionTools so each call carries a client-generated request ID; the service layer — not the agent — enforces exactly-once effects.

DISTRIBUTED-SYSTEMS ANALOGY

PUT vs. POST. Treat every agent-triggered action like a call from an unreliable client, because that's exactly what it is.

RELIABILITY PATTERN 3 OF 5

3

Circuit Breakers, Timeouts & Retries

THE PATTERN

Every tool call gets a timeout, a bounded retry budget with backoff and jitter, and a circuit breaker that opens after repeated failures.

GEMINI + ADK IN PRACTICE

ADK: before_tool_callback / after_tool_callback wraps each FunctionTool with breaker state; an open breaker returns a safe fallback instead of blocking the whole agent.

DISTRIBUTED-SYSTEMS ANALOGY

The same resilience stack you'd put around any downstream microservice call — Gemini's plan is just another unreliable caller.

RELIABILITY PATTERN 4 OF 5

4

Evaluator / Checker Agents

THE PATTERN

A separate agent, or a deterministic rule set, reviews the primary agent's proposed output or action before it is ever treated as final.

GEMINI + ADK IN PRACTICE

ADK: a lightweight LlmAgent or rule-based checker sits between the planning agent and the policy gate, scoring the proposed action against explicit constraints.

DISTRIBUTED-SYSTEMS ANALOGY

Code review, but for agent actions — a second, independent check before anything ships to production.

RELIABILITY PATTERN 5 OF 5

5

Human Approval for High-Impact Actions

THE PATTERN

Anything with real blast radius — money movement, data deletion, production changes — routes through a human before it executes.

GEMINI + ADK IN PRACTICE

ADK: the policy gate classifies action risk; low-risk actions auto-execute, high-risk actions pause the agent and wait on an explicit human sign-off.

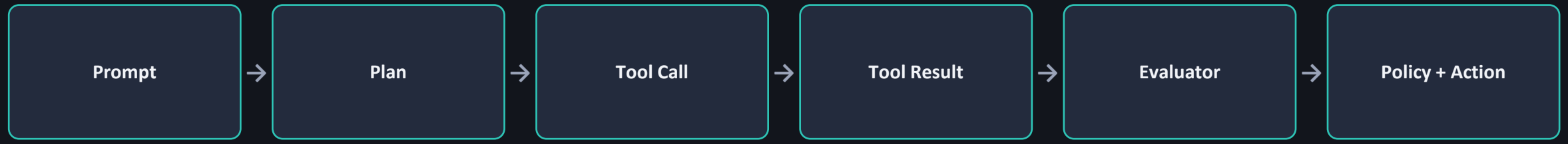
DISTRIBUTED-SYSTEMS ANALOGY

The deploy-approval gate in your CI/CD pipeline — automation earns more autonomy over time, it is never granted all of it at once.

OBSERVABILITY

Trace the entire agent trajectory

If you can't reconstruct *why* the agent did what it did, you cannot run it in production — you can only hope at it.



ONE TRACE ID PER SESSION · ONE SPAN PER AGENT OR TOOL CALL

HOW ADK GETS YOU THERE

- before/after_model and before/after_tool callbacks emit structured events
- Every Gemini reasoning step becomes a span, not just a log line
- Feed spans to an OpenTelemetry-compatible backend

THE DISTRIBUTED-TRACING PARALLEL

You wouldn't run a microservice mesh without distributed tracing across service boundaries. An agent pipeline has exactly the same shape — treat it exactly the same way.

PRODUCTION READINESS

The reliability checklist

Before an agent touches a real system, real customers, or real money — how many of these can you check off?

- ✓ Bounded iterations
- ✓ Tool schemas
- ✓ Idempotency
- ✓ Timeouts
- ✓ Retry budgets
- ✓ Permission scopes
- ✓ State validation
- ✓ Evaluator
- ✓ Audit trail
- ✓ Fallback
- ✓ Human approval
- ✓ Kill switch

Don’t ask whether your agent works.

Ask how it fails.

Engineer agents as unreliable distributed systems — bounded, observable, and recoverable — and “how it fails” stops being a scary question.

Tejas Pravinbhai Patel

Senior Software Development Engineer, Amazon · tejas.patel@ieee.org · linkedin.com/in/tejas5038

DevFest Troy '26 · HVTech · October 24, 2026