Loop Engineering

Building Autonomous Systems That Prompt Themselves

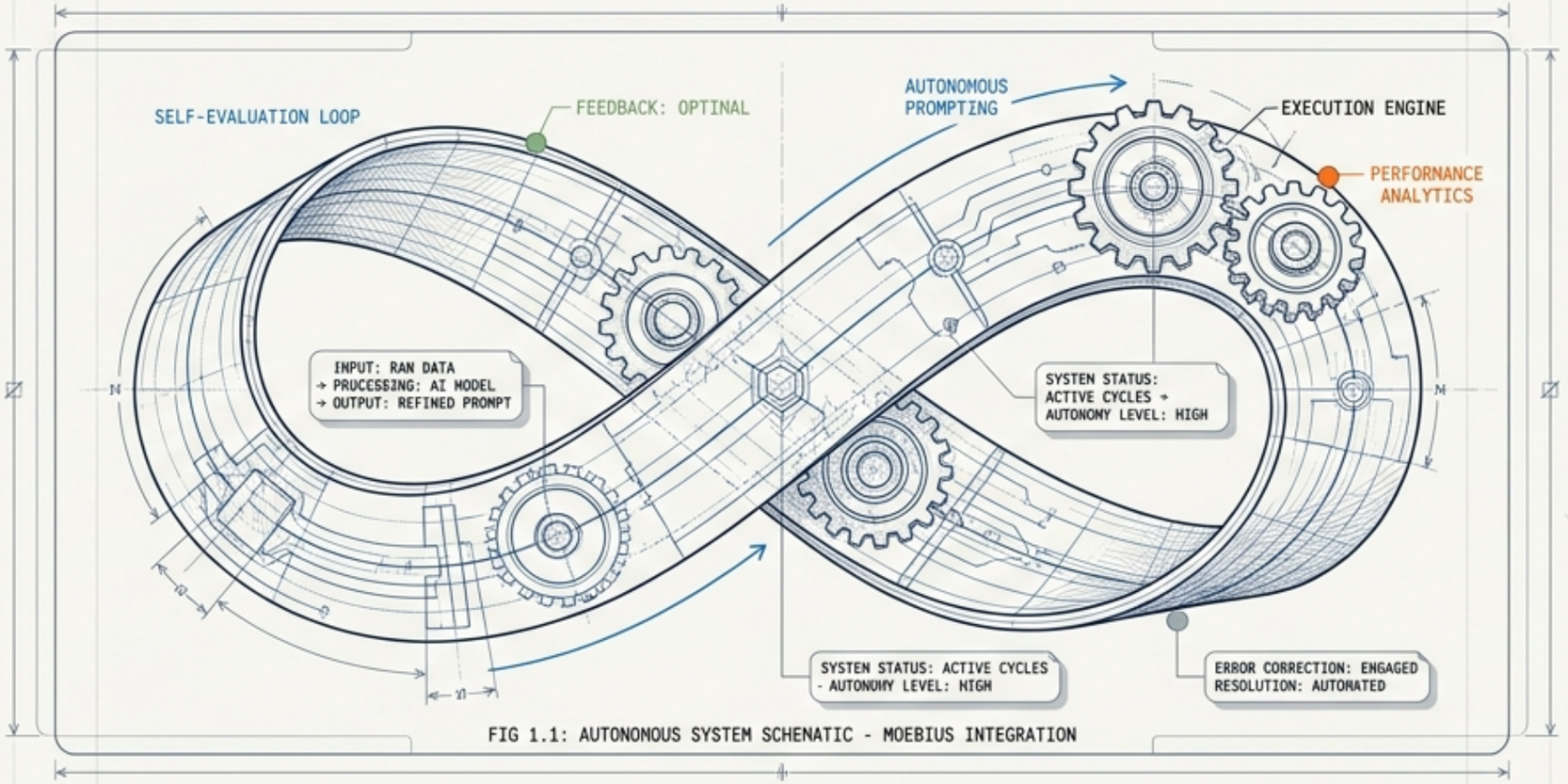
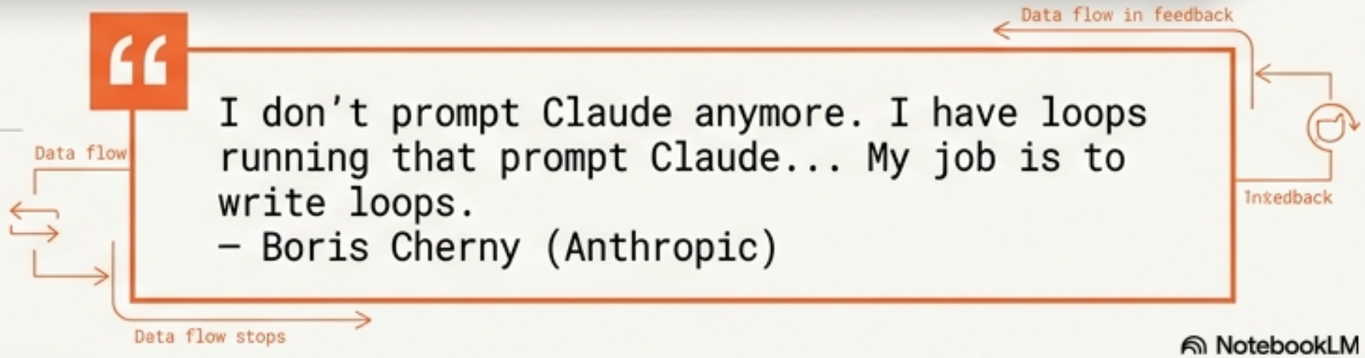


FIG 1.1: AUTONOMOUS SYSTEM SCHENATIC - MOEBIUS INTEGRATION

SYS_VERSION: 2026.1 | DEPLOYMENT_MODE: AUTONOMOUS | AUTH: VERIFIED

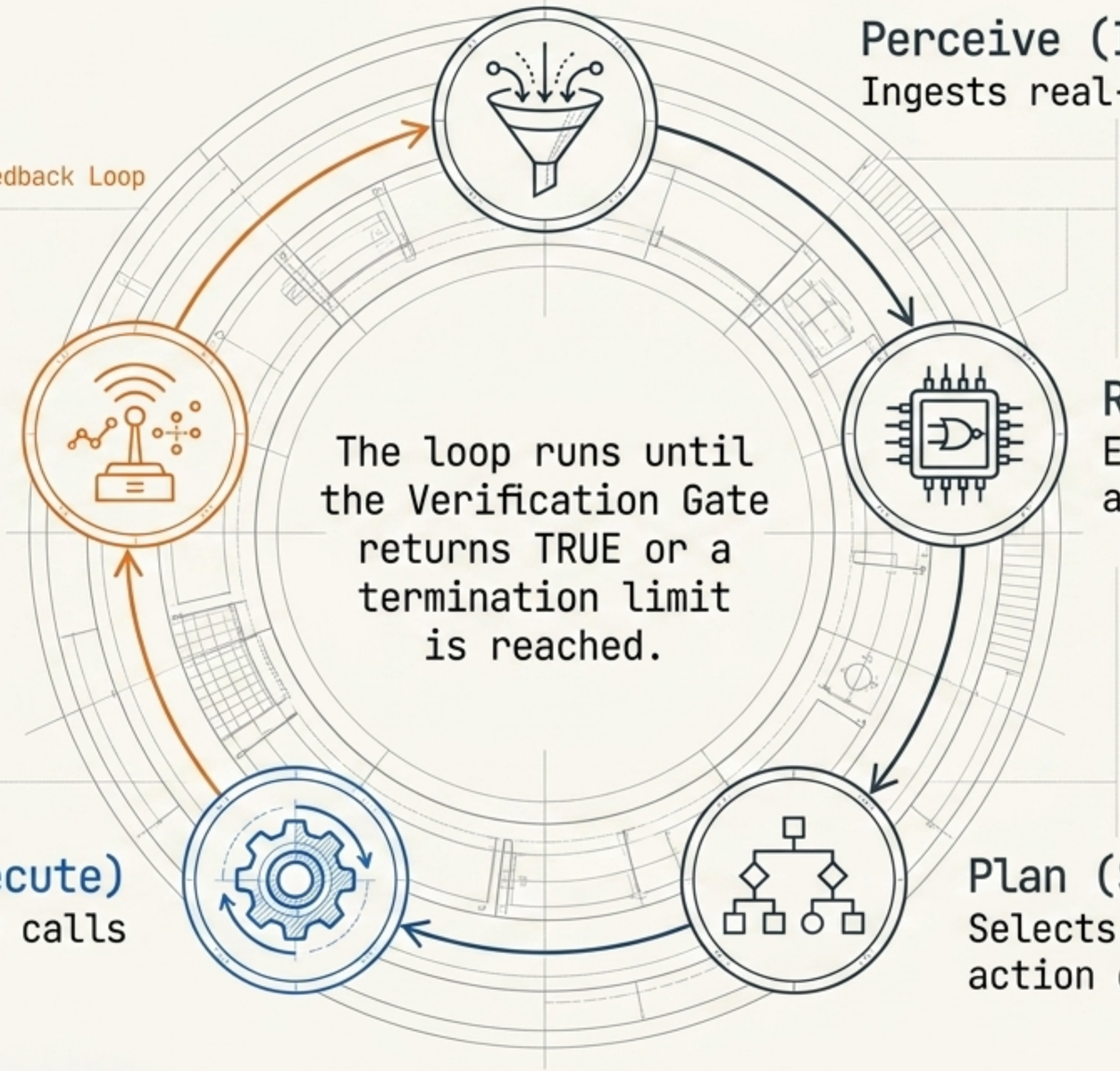
The Evolution of Work: From Manual Operation to Autonomous Loops

Dimension	One-Shot Prompting	Traditional Automations	Agentic Loops
Execution Trigger	Human typing	Fixed event	Event, Schedule, or Human
Pathing	Linear	Pre-defined recipe	Adaptive evaluation
Adaptability	None	Breaks on edge cases	Adjusts based on feedback
Stopping Condition	Single output	End of script	Verifiable goal met
Human Role	Micro-manager	Builder	System Designer



> I don't prompt Claude anymore. I have loops running that prompt Claude... My job is to write loops.
> – Boris Cherny (Anthropic)

The Anatomy of an Agentic Loop: The ReAct Circuit



Perceive (Ingest)
Ingests real-world signals

Reason (Evaluate)
Evaluates progress against goal

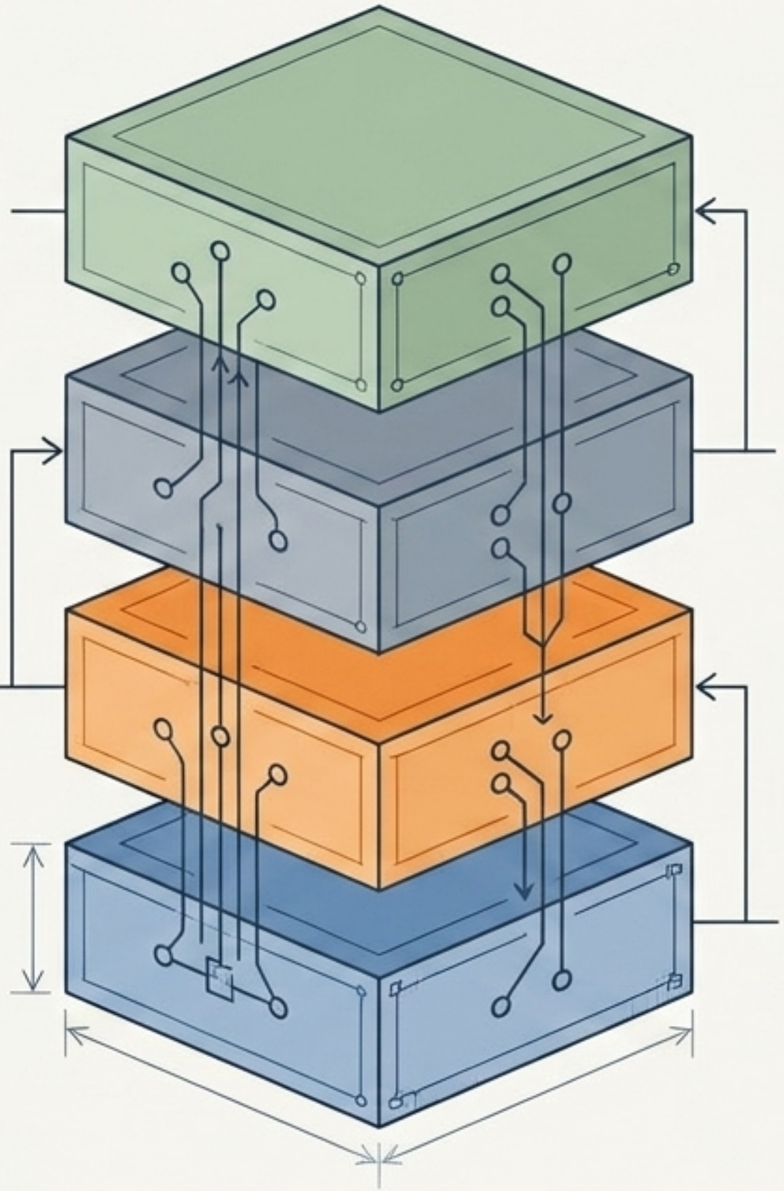
Plan (Strategize)
Selects immediate action or sub-tasks

Act (Execute)
Executes real-world calls

Observe (Feedback)
Captures outcome, updates context

The 4-Level Stack: Encapsulating Complexity

Level 4: Hill Climbing Loop
Automates system improvement. Rewrites harness config based on production traces.

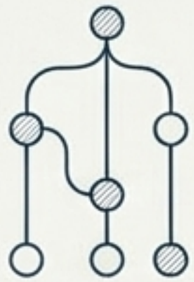


Level 3: Event-Driven Loop
Background execution triggered by schedules, webhooks, or channels.

Level 2: Verification Loop
Grader checks output against rubric. Routes feedback on failure.

Level 1: The Agent Loop
Model plans and calls tools repeatedly in isolated environment.

Core Environmental Primitives: The System Requirements



Worktrees (Isolation)

Prevents parallel agents from colliding. Gives each sub-agent a clean checkout.

Skills (Context)

Codifies project knowledge in SKILL.md so the loop doesn’t re-derive context from zero.

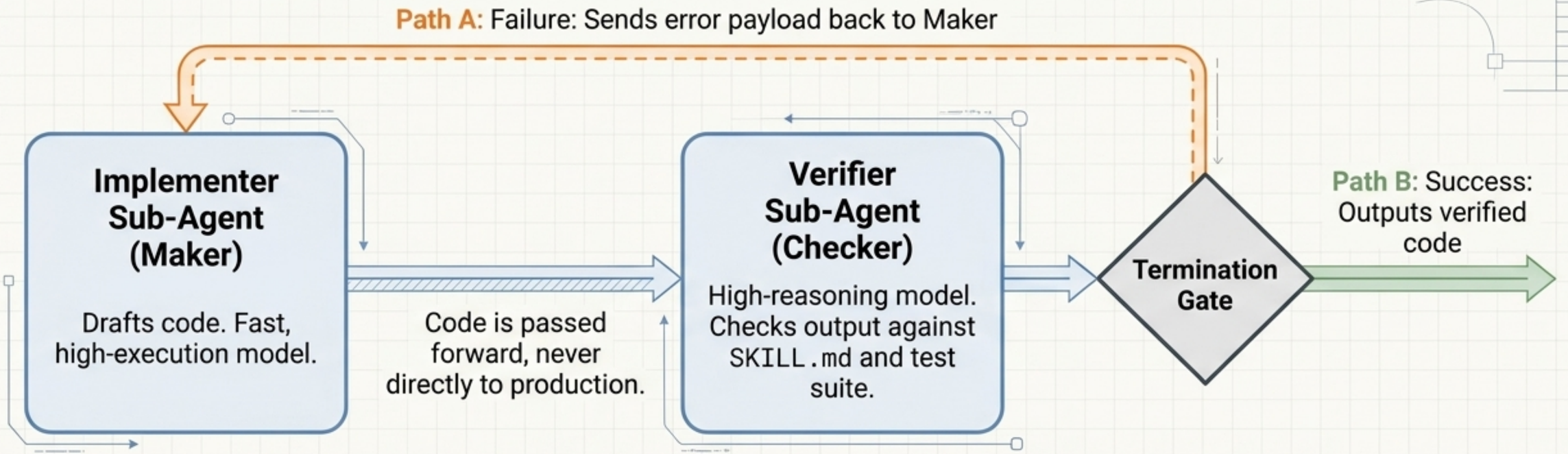
Connectors (Reach)

MCP servers that allow the loop to read issue trackers, databases, or post updates.

State (Memory)

The agent forgets; the repo doesn’t. Tracks done/next state between runs.

Adversarial Code Review: The Maker vs. Checker Architecture



Key Insight Box

A loop running unattended is a loop making mistakes unattended. Splitting the maker from the checker is the only way to trust the exit condition.

Tactical Implementation: Giving the Engine a Compass

```
> /goal Achieve 85% test coverage as reported by pytest-cov
```

The Compass

Provides a verifiable, scoped, metric-based completion criterion.

```
> /loop run pytest test/auth --fix-failures
```

The Engine

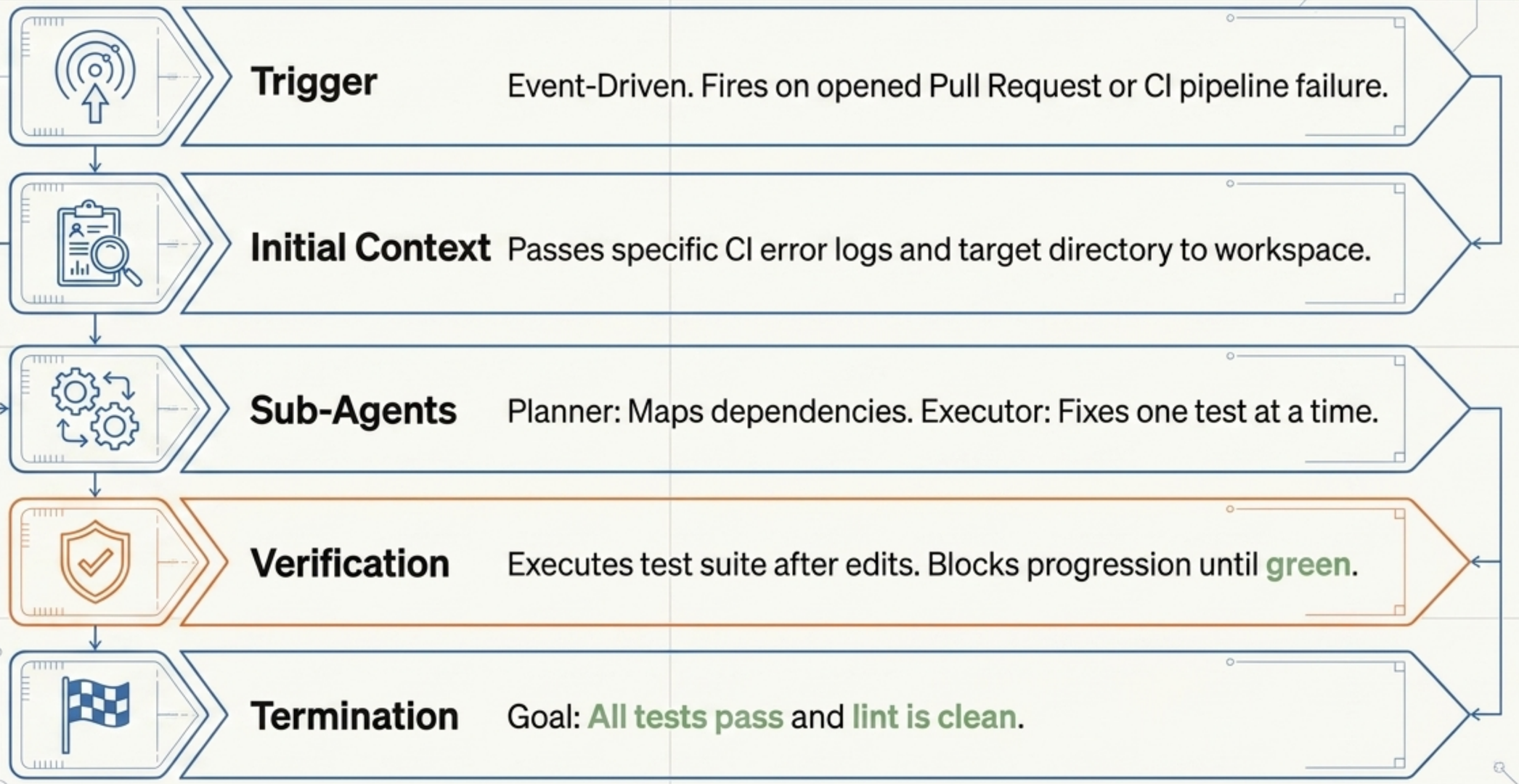
Forces the agent to cycle through assess-act-verify until the goal is met.

```
> claude -p '/loop fix linting errors' > loop.log 2>&1
```

Autonomous Mode

Removes the human entirely. Enables persistent background workers via OS schedulers.

Blueprint I: Test-Driven Bug Fixing



Blueprint II: Autonomous Documentation Generation

Trigger Schedule-Based. Runs nightly via a cron job on main branch.

Initial Context Reads SKILL.md for brand voice, formatting, and target audience.

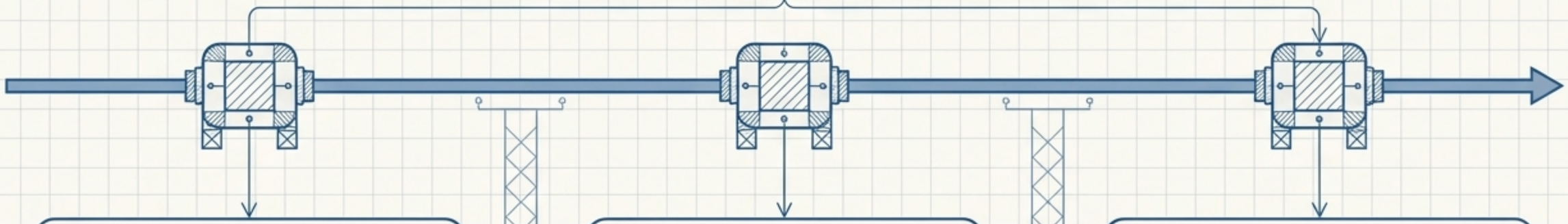
Sub-Agents Researcher: Audits new code. Writer: Generates Markdown docs.

Verification Semantic review against brand context. Scripts verify markdown links.

Termination Goal: Every exported function has link-verified documentation.

Blueprint III: Complex Stack Migration

Trigger: Human-Initiated execution of Phase 1.



Checkpoint 1: Auth Migration

Action: /loop migrate Supabase Auth to Clerk

Termination: Test suite confirms login success. System halts.

Checkpoint 2: Data Layer

Action: /loop swap NextJS API for Convex mutations

Termination: SKILL.md schema rules met. DB queries typed.

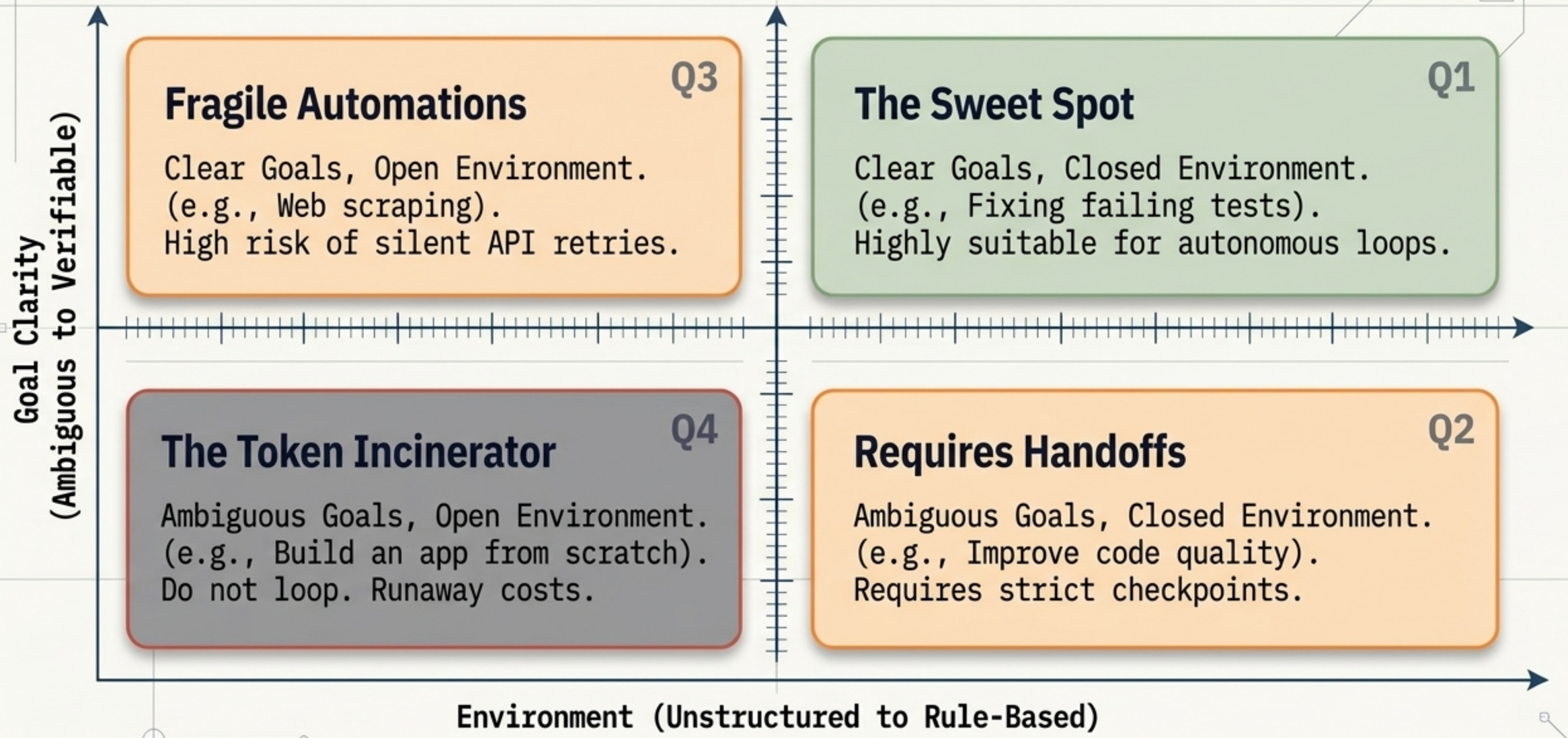
Checkpoint 3: Framework Swap

Action: /loop migrate NextJS pages to React/Vite

Termination: Vite build succeeds with zero errors.

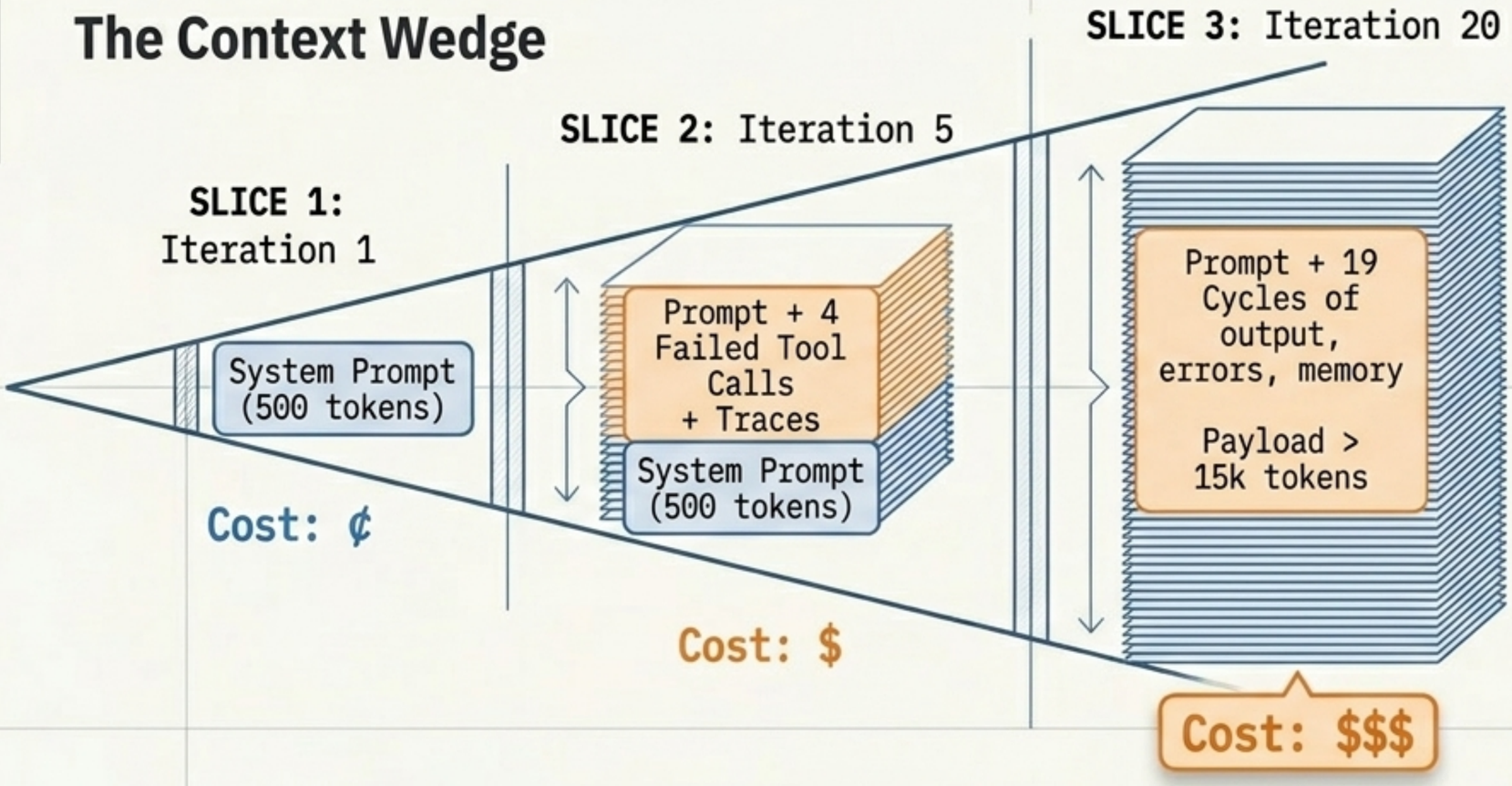
⚠ Do not use a single goal for a full migration. Break it into verifiable checkpoints to preserve context and ensure audibility.

Managing Unknowns: The Task Suitability Diagnostic



The Financial Physics of Loops: Context Carryover

The Context Wedge



An unmonitored codebase refactor loop can consume over 180,000 tokens per session, leading to massive runaway API bills.

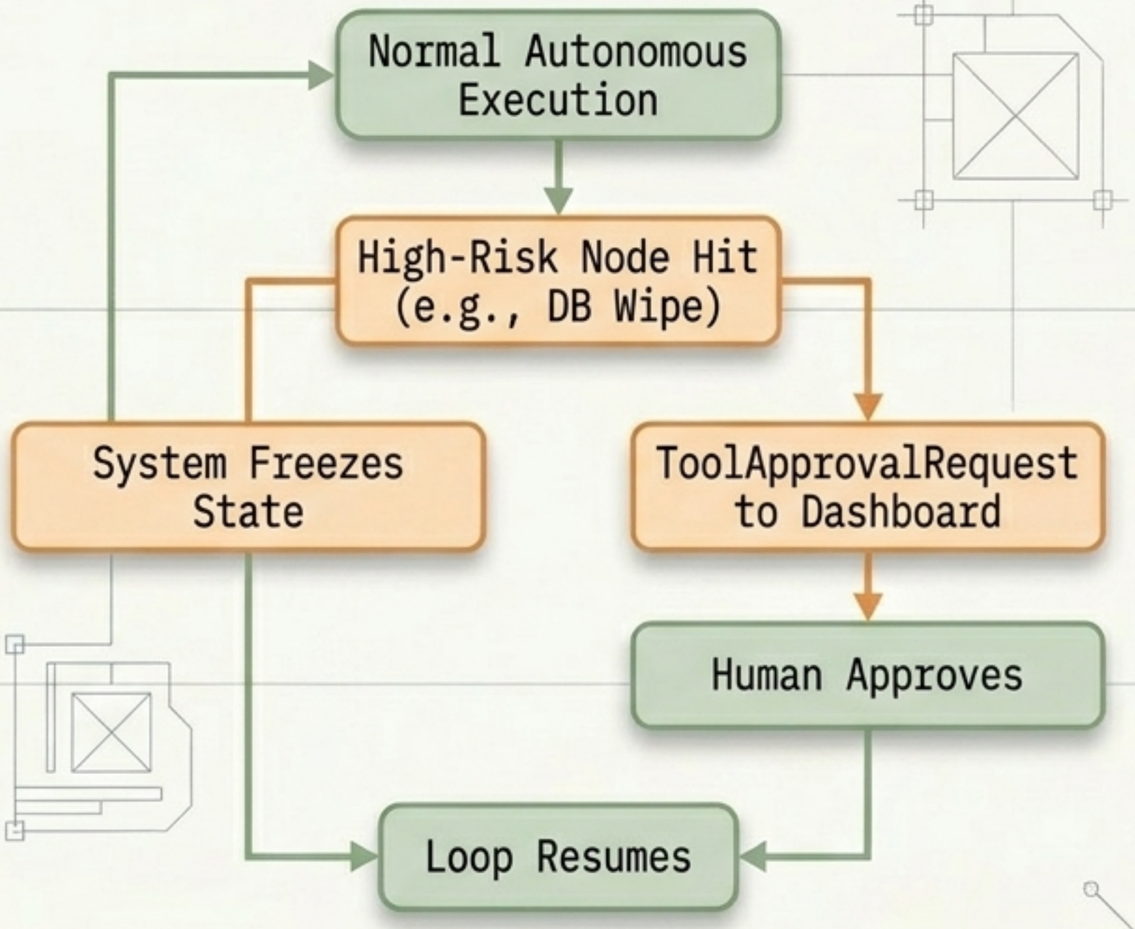
Because LLMs are stateless, every iteration repays the cost of the entire accumulated history.

Cost Governance & Dynamic Interrupts

Risk Mitigation Table

Risk Vector	Technical Mechanism	Mitigation Strategy
Runaway Loops	Semantic Loop Breakers	Halts execution if agent circles the same semantic state.
Silent Retries	Token-Velocity Circuit Breakers	Trips if consumption exceeds 10k tokens/minute.
Budget Exhaustion	Code-Based Guards	Independent checkBefore() budget limits.

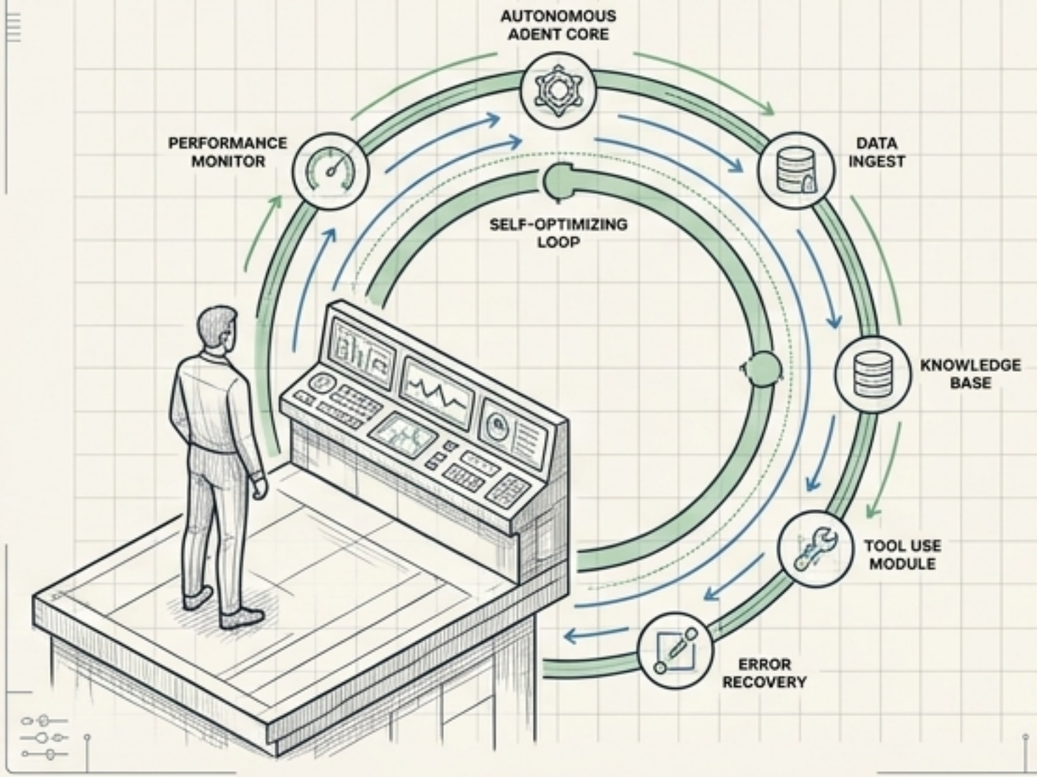
Dynamic Interrupt Flowchart



Synthesis: The System is the Asset, Not the Prompt

Cognitive Labor

Cognitive Capital



The ultimate goal of Loop Engineering is replacing yourself as the person who prompts the agent. We are moving from cognitive labor (typing prompts) to cognitive capital allocation (designing autonomous systems). The leverage belongs to the engineer.

Glossary of Loop Engineering

ReAct Pattern	Architecture alternating Reasoning steps with Actions to prevent blind retries.
Sub-Agent	Isolated LLM worker deployed within a loop.
Worktree	Isolated Git directory ensuring parallel agents do not collide.
Context Carryover	Escalating token cost of passing historical logs into new iterations.
Deterministic Goal	Clear, objectively verifiable exit condition.
Semantic Loop Breaker	Guardrail that halts an agent repeating identical thoughts.
HOTL (Human-on-the-Loop)	Oversight model where humans intervene only for high-risk exceptions.
Dynamic Interrupt	System freeze that saves state and requests human approval.