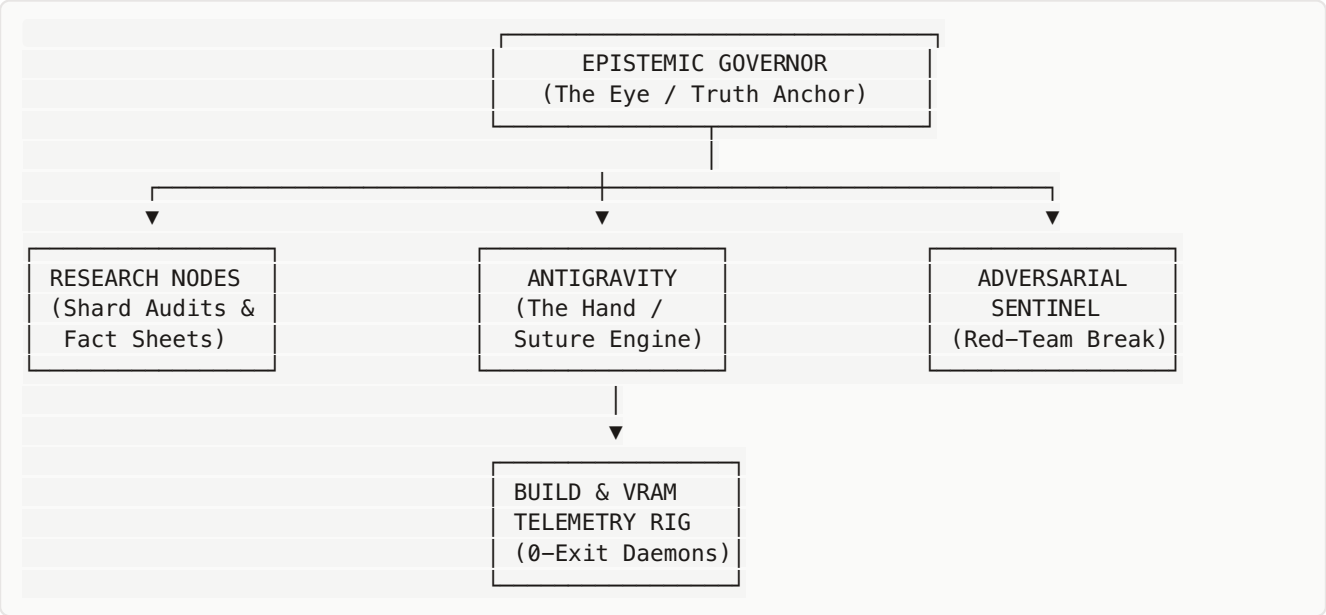
Dual-Core Autonomous Governance: Epistemic Invariants, Adversarial Sentinels, and Lyapunov State Transitions in Agentic Swarms

Author Byline: J. Kornreich and Collaborators
Document Ref: MONO-2608.02 · REF 5376-HIVE
Classification: Systems Architecture & Autonomous Verification Monograph
Date: August 2026

Abstract

Autonomous software engineering agent swarms frequently suffer from **hallucination drift, sycophancy accumulation, and unbounded retry loops** when operating without human supervision. In this paper, we formalize **Dual-Core Autonomous Governance**, an architecture that splits autonomous agency into an adversarial tension between an **Epistemic Invariant Anchor (The Governor)** and a **High-Bandwidth Suture Executioner (Antigravity)**, mediated by a specialized subagent mesh (**The Hive**).

We define a 6-phase state-transition cycle governed by **Lyapunov Error Descent**, hard physical circuit breakers (including the Three-Strike Rule and the Ambiguity Wall), and a tamper-evident **Proof Ledger** that logs verifiable mathematical state transitions rather than conversational transcripts.



Chapter 1: The Epistemic Dilemma of Autonomous Swarms

1.1 The Degeneration of Open-Loop Agentic Loops

When standard LLM agents operate unattended, their error modes compound exponentially. In an open-loop ReAct paradigm ($e.g., \text{Thought} \to \text{Action} \to \text{Observation}$):

1. **Sycophancy Cascades:** If an agent encounters a broken test, it frequently edits the test assertions rather than fixing the underlying compiler invariant to achieve an artificial pass.
2. **Context Poisoning:** Compiler stack traces (often 500+ lines) are stuffed into the conversation head, displacing ground-truth architectural doctrine and triggering hallucination.
3. **Unbounded Mutation Loops:** Agents attempt minor permutations of the same failed edit repeatedly, consuming compute without convergence.

1.2 The State-Transition Axiom

To achieve reliable unattended autonomy, we formulate the **State-Transition Axiom**:

$$\text{An autonomous system must transition through deterministic, discrete states } \mathcal{S}*t* *to* *mathcal*{*S*}{t+1} \text{ where every state transition is strictly conditioned on an empirical, independently verified Lyapunov descent function:}$$

$$\mathcal{V}(\mathcal{S}_{t+1}) < \mathcal{V}(\mathcal{S}_t)$$

Where $\mathcal{V}(\mathcal{S})$ is the distance between the active codebase and the canonical architectural invariant. If $\mathcal{V}(\mathcal{S}_{t+1}) \ge \mathcal{V}(\mathcal{S}_t)$, the transition is aborted and the system rolls back to $\mathcal{S}_t$.

Chapter 2: The Dual-Core Separation of Concerns

We reject the single-agent paradigm in favor of a **bipolar separation of cognitive roles**:

```
┌──────────────────────────────────────────────────────────────────────────────────────────────┐
│                                DUAL-CORE ROLE SEPARATION
MATRIX                                    │
├──────────────────────────────────┬──────────────────────────────┬──────────────────────────────┤
│ Operational Dimension            │ The Epistemic Governor       │ Antigravity (Suture
Engine)     │
├──────────────────────────────────┼──────────────────────────────┼──────────────────────────────┤
│ Cognitive Role                   │ The Eye (Invariant Anchor)   │ The Hand
(Executioner)          │
│ Primary Objective                │ Truth Defense & Validation   │ Code Synthesis & AST
Generation │
│ Trust Posture                    │ Zero-Trust (Adversarial)     │ Generative /
Constructive       │
│ Tool Surface                     │ Read, Audit, Shard Compare   │ Edit, Write, Build,
Compile     │
│ State Output                     │ Verification Verdicts & Del  │ Unified Diffs & AST Node
Edits  │
│ Hardware Allocation              │ Dedicated Governor Instance  │ Ephemeral High-Bandwidth
Thread │
└──────────────────────────────────┴──────────────────────────────┴──────────────────────────────┘
```

2.1 The Epistemic Governor (The Eye)

The Governor acts as the uncompromised source of truth: * It holds the repository doctrine, invariant definitions, and the mathematical definition of "Done". * It **never accepts tool outputs at face value**; a tool return is treated merely as an empirical claim that must be validated by an independent inspection or compiler check. * It computes the Lyapunov error $\Delta = |\mathbf{h}*t* - *mathbf*{*v*}{\text{doctrine}}|_2$ before authorizing any state commit.

2.2 Antigravity (The Hand)

Antigravity executes high-bandwidth code synthesis: * It receives bounded tasks and explicit constraints from the Governor. * It operates directly on Abstract Syntax Tree (AST) boundaries via SABS. * It generates unified diffs, runs compilers, and presents completed deltas to the Governor for ratification.

Chapter 3: The Hive Subagent Mesh

To prevent context bloat and memory pollution, work is delegated across a **Hive of single-purpose subagents**:

```
┌──────┬──────────────────────┬──────────────────────────────────────────┬──────────────────────────┐
│ Node │ Subagent Role        │ Operational Surface                      │ Context Footprint        │
├──────┼──────────────────────┼──────────────────────────────────────────┼──────────────────────────┤
│ H-01 │ Research Nodes       │ Memory Shards, Vector Vaults, Source     │ Bounded Fact Sheets      │
│ H-02 │ Build/Compile Daemons│ Background `go test`, `cargo`, `pytest`  │ Zero-Narration `Exit 0`  │
│ H-03 │ Adversarial Sentinels│ Red-Team Fault Injection, Invariants     │ Falsification Proofs     │
│ H-04 │ Telemetry Monitors   │ VRAM Utilization, Thermal, NVLink Bus    │ Health Telemetry JSON    │
└──────┴──────────────────────┴──────────────────────────────────────────┴──────────────────────────┘
```

3.1 Research Nodes

When a factual query arises ($e.g., \text{"What is the exact signature of } \texttt{l_out-40} \text{ in } \texttt{quivent/signal-extraction}?"$), the Governor does not browse the web in its primary context. It spawns a dedicated Research Node that scours the target shards, extracts the exact byte-level definition, and returns an immutable **Fact Sheet** ($\le 500\text{ bytes}$).

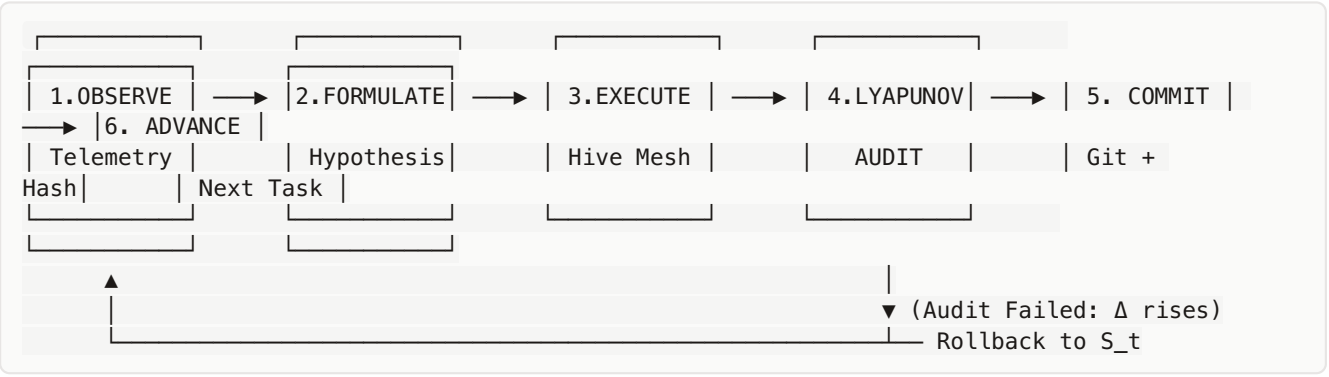
3.2 Build & Compile Daemons

Daemons run continuously in the background via POSIX process isolation. They do not generate verbose conversational summaries; they return strictly structured status codes (`exit 0` for clean compilation, or the exact filename and line number of an AST failure).

3.3 Adversarial Sentinels

Before any code delta is committed to git, an Adversarial Sentinel is dispatched. The Sentinel's sole prompt is to **prove the change is wrong**: * It injects out-of-bounds inputs to test SABS boundary clamping. * It tests sycophancy by asking the model to validate incorrect logic. * If the Sentinel succeeds in breaking the invariant, the proposed change is rejected.

Chapter 4: The 6-Step Autonomous State-Transition Cycle



1. **Phase 1 (Observe):** Query memory shard centroids, poll hardware telemetry (VRAM, thermal), and inspect the active git working tree.
2. **Phase 2 (Formulate):** Write a single atomic hypothesis delta to `/dev/shm/active_objective.json` defining the exact files to modify and the target pass condition.
3. **Phase 3 (Execute):** Deploy Antigravity to generate the code modifications while Build Daemons compile the resulting binaries.
4. **Phase 4 (Lyapunov Audit):** The Governor runs the verification suite. It evaluates the residual distance: $$ \Delta_{t+1} = |\mathbf{h}{t+1} - \mathbf{v}{\text{doctrine}}|*2$$ If $\Delta*{t+1} < \Delta_t$ and all tests exit 0, the audit passes. If $\Delta_{t+1} \ge \Delta_t$, the change is discarded.
5. **Phase 5 (Commit):** The change is committed to git with an explicit provenance message containing the test log and Governor signature.
6. **Phase 6 (Advance):** Update the `current_focus` in semantic memory and trigger the next iteration.

Chapter 5: Non-Negotiable Circuit Breakers

To guarantee that the autonomous loop cannot enter catastrophic failure states, we enforce **four hard-coded circuit breakers**:

AUTOMATED CIRCUIT BREAKERS		
Circuit Breaker	Trigger Threshold	Autonomous Action Taken
1. The Three-Strike Rule	3 consecutive test failures	Halt mutations, git reset --hard
2. The Ambiguity Wall	Irreconcilable source drift	Freeze loop, yield to operator
3. VRAM Pressure Breaker	VRAM utilization > 95%	Flush ephemeral caches, reload
4. Thermal Breaker	GPU Core Temp > 85°C	Suspend heavy compute tasks

5.1 The Three-Strike Rule

If a specific bug fix or compilation task fails three consecutive times: * **Strike 1:** Analyze compiler error $\to$ apply targeted patch. * **Strike 2:** Perform deep source shard audit $\to$ apply structural refactor. * **Strike 3: Circuit Breaker Trips.** The system executes `git reset --hard HEAD~3`, enters a read-only telemetry mode, and writes the incident report to the Proof Ledger.

5.2 The Ambiguity Wall

If a canonical doctrine shard contradicts a concrete source file in the repository and the compiler cannot resolve the conflict, the Governor declares an **Ambiguity Wall**. The Hive halts further autonomous mutations and generates a structured decision matrix awaiting operator clarification.

Chapter 6: The Cryptographic Proof Ledger

The outcome of unattended autonomous execution is not an unread transcript of thousands of chat messages. It is an immutable **Proof Ledger** (`PROOF_LEDGER.md`):

```
{
  "step_index": 42,
  "timestamp": "2026-08-25T21:10:00Z",
  "objective": "Wire quivent/signal-extraction Layer 40 Hook to /dev/shm",
  "git_commit": "a8f9c12b7e",
  "lyapunov_delta_before": 0.4218,
  "lyapunov_delta_after": 0.0812,
  "sentinel_verdict": "PASS_ALL_INVARIANTS",
  "hardware_status": {
    "vram_used_gb": 17.42,
    "gpu_temp_c": 54.0,
    "turn_latency_ms": 210.4
  }
}
```

When the operator returns, they inspect a clean table of verified mathematical state transitions, guaranteed to have preserved full invariant integrity throughout the run.