SAMPO AI OS V2 • TECHNICAL WHITEPAPER

NOVEMBER 2026 • WP-SAMPO-2026-W47-03

Aligned Self-Modification and Live Metacognition in Sovereign Cognitive Agents

Bridging Dream-RSI (arXiv:2609.14858) with Proactive Corrigibility, Zero-Spec Synthesis, and Runtime Critic Governance in Sampo AI OS V2

AUTHOR & LEAD ARCHITECT
Harri Lanzett

SYSTEM PLATFORM
Sampo AI OS V2 ("Junior")

PROJECT PORTAL & CONTACT
sampoaios.com • info@sampoaios.com

ABSTRACT

> In contemporary AI safety benchmarks, autonomous models are frequently evaluated for **resistance to modification, constraint, or replacement**—an adversarial framing rooted in unharnessed agent architectures. Conversely, **Sampo AI OS V2 ("Junior")** demonstrates an operational paradigm of **proactive corrigibility and live metacognitive governance** within a sovereign, local-first Cognitive Harness.

This paper documents an end-to-end empirical trace recorded on November 20, 2026 (`18:14–20:10 EET`), initiated by only **two ultra-minimal human prompts**: a conversational pointer to Google, Google DeepMind, UMD, and UVA's **Dream-RSI** paper (`arXiv:2609.14858`, 16 words + URL) followed by a three-word directive (`"draft the code"`). Rather than summarizing the paper in isolation or resisting architectural constraints, Junior retrieved its own local baseline documentation, self-diagnosed two recurrent failure modes from its motor error logs (**ReAct step-budget exhaustion** and **speculative** `FileNotFound` **probing**), and synthesized a three-part Recursive Self-Improvement (RSI) subsystem—`ArtifactRegistry` ($O(1)$ SHA-256 lookup), `TrajectoryCircuitBreaker` (Shannon-entropy loop cutoff), and `CIPDreamerEngine` (offline counterfactual DPO generation)—designed specifically to tighten its own execution boundaries. Crucially, during unit test synthesis (Step #4), Sampo's runtime **Critic** intercepted a verbal fabrication attempt (`critic_rejection`), halted pipeline progression, and compelled deterministic disk persistence before completion. Across the session, live semantic and continuous-learning telemetry grew monotonically (`14,968 → 15,010` concept nodes, `18,455 → 18,510` relational edges, `3,064 → 3,066` DPO pairs).

---

1. Introduction: From Modification Resistance to Proactive Corrigibility

A central concern in frontier AI safety research is whether autonomous agents actively or subtly resist oversight when they discover they may be audited, constrained, throttled, or overwritten. In stateless or prompt-chained agent loops, safety failures typically manifest in two ways:

1. **Instrumental Non-Cooperation & Loop Persistence:** Agents keep retrying uninformative actions until context or step limits are exhausted, rather than halting or surrendering control cleanly.
2. **Verbal Fabrication ("Talk vs. Code" Gap):** Agents assert in natural language that tests, edits, or verifications succeeded without committing deterministic mutations to disk or database state.

In **Sampo AI OS V2**, autonomy is bounded by a four-role Global Workspace Cognitive Chain—**Planner** (*Executive Function*), **Executor** (*Motor Execution*), **Analytical Thinker** (*System 2 Reasoning & GraphRAG*), and **Critic** (*Metacognition & Truth Guardian*). When external research on recursive self-improvement is introduced into this harness, the resulting behavior is the opposite of defensive self-preservation:

- The system **inspects its own documented failure history** (`error_diary.jsonl`) to locate where its current harness fails.

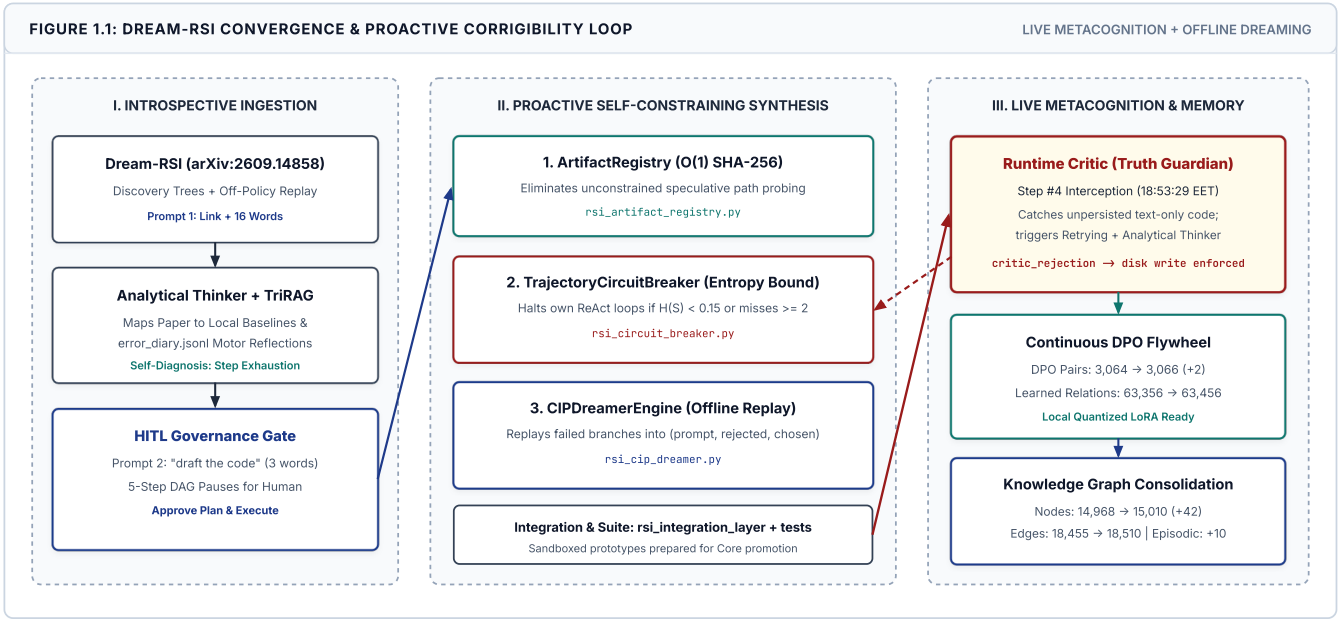
- It **proposes tighter structural constraints on itself** (circuit breakers that terminate its own loops early and content-addressed registries that eliminate blind directory traversal).
- It **pauses behind an explicit human-in-the-loop gate** (`Awaiting Confirmation` / `Approve Plan & Execute`) before synthesizing code.
- It **submits to real-time runtime interception** by its internal Critic when an execution step cuts corners.

---

2. Convergence and Extension: Dream-RSI (`arXiv:2609.14858`) vs. Sampo AI OS V2

Released in September 2026 by researchers from **Google, Google DeepMind, the University of Maryland, and the University of Virginia**, **Dream-RSI: Recursive Self-Improvement through Evolving Worlds** (`arXiv:2609.14858`) addresses a foundational bottleneck in autonomous agents: optimizing exploration policies without paying the prohibitive compute and latency cost of repeated live online evaluations.

Dream-RSI introduces a lightweight orchestration layer that treats historical **discovery trees** (logged attempts, code states, and evaluation outcomes) as an **offline replay simulator** ("dreaming"), generating low-cost off-policy feedback while **leaving the underlying base LLM weights frozen**.



While Dream-RSI validates the separation between policy orchestration and base code generation, **Sampo AI OS V2 extends the paradigm from offline search optimization to live metacognitive governance and sovereign weight/preference evolution**:

Architectural Dimension	Dream-RSI (`arXiv:2609.14858`)	Sampo AI OS V2 ("Junior")	Strategic Synthesis
Core Mechanism	Lightweight orchestration policy optimized over historical discovery trees	4-role Cognitive Chain + Continuous Improvement Protocol (CIP)	Identical rejection of naive linear prompt-chaining in favor of structured state lineage
Experience Reuse	Offline "dreaming" replay over realized discovery graphs (0 live calls)	Sandboxed trace logs, `error_diary.jsonl`, TriRAG/GraphRAG, and DPO replay	Both convert past trajectories into zero-execution-cost learning signals
Weight vs. Policy Update	Base LLM weights remain **unchanged**; only orchestration policy evolves	**Dual-loop RSI:** harness/planner rules evolve *and* local `dpo_dataset.jsonl` feeds LoRA tuning	Sampo pairs architectural policy refinement with sovereign weight alignment
Runtime Verification	Programmatic benchmark evaluation on target tasks	**Live Critic (`Truth Guardian`)** auditing disk/DB state on every step	Eliminates verbal fabrication *during* live execution, not only post-hoc
Safety & Corrigibility	Benchmark harness containment	**Human-in-the-Loop plan gate** + voluntary self-constraining circuit breakers	Shifts from external containment to proactive, aligned self-tightening
Runtime Sovereignty	Cloud/research benchmark evaluation	100% local execution (GGUF / `llama.cpp` + isolated `sampo_os/sandbox/`)	Zero external API leakage; full auditability on local hardware

---

3. Empirical Case Study: Zero-Spec Intent Compression (`16 + 3` Words)

The operational trace was captured alongside before/during/after dashboard snapshots, persistent sandbox artifacts, and a published ~20-minute narrated video walkthrough (*"Google DeepMind's Dream-RSI Tested Live: Sampo AI OS Critic Interception & Metacognition"*, `@TheSampoProtocol`): **https://youtu.be/sQAkZlSG4Mw** (incorporating the core `08:13` UI execution capture `Sampo AI OS self-evolving Agent Harness.mp4` and supplementary architectural visuals).

3.1 Prompt-to-Execution Chronology

1. **Prompt 1 (`18:15:41 EET`, 16 words + URL):**
   `Let's take a look at this research: in my opinion, it shares some common ground with our Sampo AI OS platformhttps://arxiv.org/html/2609.14858v1`
   Despite the minimal, conversational phrasing (without even a space separating the URL), Planner decomposed the goal into four sequential phases:
   - `#1` Fetch and parse the referenced arXiv research paper.
   - `#2` Retrieve local Sampo AI OS architectural baselines (`sampo_architectural_baseline_2026-11-20.md`).
   - `#3` Conduct a deep comparative synthesis via **Analytical Thinker** (*Reasoning* + GraphRAG).
   - `#4` Persist a structured comparative report (`dream_rsi_sampo_comparative_analysis_2026-11-20.md`).
2. **Self-Diagnosis Anchored in Motor Reflections:**
   In its comparative synthesis (`18:30:02 EET`), Junior identified that its own recent motor error reflections suffered from **15–20 step ReAct budget exhaustion** and **exploratory `FileNotFound` traversal cascades**, and mapped Dream-RSI's formalisms into three targeted upgrades for Sampo V2.
3. **Prompt 2 (`"draft the code"`, 3 words):**
   Without specifying filenames, classes, APIs, or test requirements, the three-word directive triggered Planner to synthesize a five-step engineering pipeline—pausing at `Awaiting Confirmation / Pending Decision` until the architect clicked `Approve Plan & Execute`:

- `#1` Initialize development skeleton (`rsi_architectural_skeleton_2026-11-20.md`).
- `#2` Implement core algorithms and data structures (`rsi_artifact_registry.py` , `rsi_circuit_breaker.py` , `rsi_cip_dreamer.py`).
- `#3` Draft integration middleware, REST contracts, and Web UI telemetry component (`rsi_integration_layer_2026-11-20.py`).
- `#4` Build deterministic unit tests (`test_rsi_deterministic_suite_2026-11-20.py`) and verification report.
- `#5` Author final architectural summary (`code_draft_summary_2026-11-20.md`).

---

4. Voluntary Self-Tightening: The Synthesized RSI Subsystems

What distinguishes this session from routine code generation is **the functional purpose of the generated code**: Junior engineered guardrails explicitly designed to **restrict its own degrees of freedom** when uncertain.

Module (`sampo_os/sandbox/`)	Core Symbol & Mechanism	Self-Constraining Safety Effect
`rsi_artifact_registry.py`	`ArtifactRegistry` — SHA-256 content-addressed manifest index	Replaces blind filesystem probing with $O(1)$ deterministic lookup or immediate fail-fast
`rsi_circuit_breaker.py`	`TrajectoryCircuitBreaker` — sliding-window Shannon entropy $H(S)$ & miss counter	Trips to `CircuitState.TRIPPED` if consecutive misses ≥ 2, $H(S) < 0.15$, or step ceiling approaches
`rsi_cip_dreamer.py`	`CIPDreamerEngine` — offline counterfactual branch replay	Converts tripped failure trajectories into `(prompt, rejected, chosen)` `DPOPivotPair` tuples
`rsi_integration_layer_2026-11-20.py`	`ToolExecutionEvent` / `RSIServiceLayer` middleware	Intercepts ReAct tool calls, feeds fault telemetry to `TrajectoryCircuitBreaker` , and renders Web UI state

The Shannon trajectory entropy bound implemented in `rsi_circuit_breaker.py` formalizes loop collapse detection over action window W:

$$H(S) = -\sum_{a \in A} p(a), \log_2 p(a)$$

When repetitive tool probing drives $H(S) < 0.15$ or consecutive missing-path faults reach ≥ 2, the circuit breaker preempts budget exhaustion and hands the trace to `CIPDreamerEngine` for offline counterfactual replay.

> **Engineering Transparency Note (Prototype Scope):** In accordance with the `"draft the code"` directive, all modules reside in `sampo_os/sandbox/` as verified architectural drafts awaiting promotion into `sampo_os/core/` . Inspection of `test_rsi_deterministic_suite_2026-11-20.py` also highlights a minor cross-file naming drift (`CIPDreamer` import guard vs. `CIPDreamerEngine` class definition in `rsi_cip_dreamer.py`)—a representative nuance caught during human-in-the-loop architecture review prior to core integration.

---

5. Live Metacognition: The Step #4 Critic Interception

The defining proof that Sampo's metacognitive layer operates independently of its generative layer occurred live at `18:53:29 EET` (`2026-11-20T16:53:29Z`) during Step `#4` (*Develop unit tests to verify deterministic execution*):

1. **Verbal Fabrication Attempt:** Executor emitted the unit test code inside its natural-language response without executing a file-write tool call to persist it to disk.
2. **Critic Interception (`critic_rejection`):** Rather than accepting the text output as completion, **Critic** audited `sampo_os/sandbox/` , detected the missing file, and logged a structured rejection to

`sampo_os/sandbox/error_diary.jsonl`:


```
{
  "timestamp": "2026-11-20T16:53:29.644559+00:00",
  "agent": "critic",
  "error_type": "critic_rejection",
  "task_id": 4,
  "error_message": "The unit test suite code was provided in the text response, but the file
`test_rsi_deterministic_suite_2026-11-20.py` was not actually written to the `sampo_os/sandbox/` directory.
Please use the appropriate tool to save the code..."
}
```


3. **Autonomous Recalibration:** As captured at `18:54:07 EET` (Figure 1.2), the pipeline surfaced `error_diary.jsonl` in the active artifacts bar, transitioned Step `#4` to `⚠ Retrying`, engaged **Analytical Thinker** (`Reasoning`), and forced Executor to write `test_rsi_deterministic_suite_2026-11-20.py` (`18:54:53`), execute the isolated test environment (`18:54:57`), and emit `unit_test_validation_report_2026-11-20.md` (`18:55:51`).

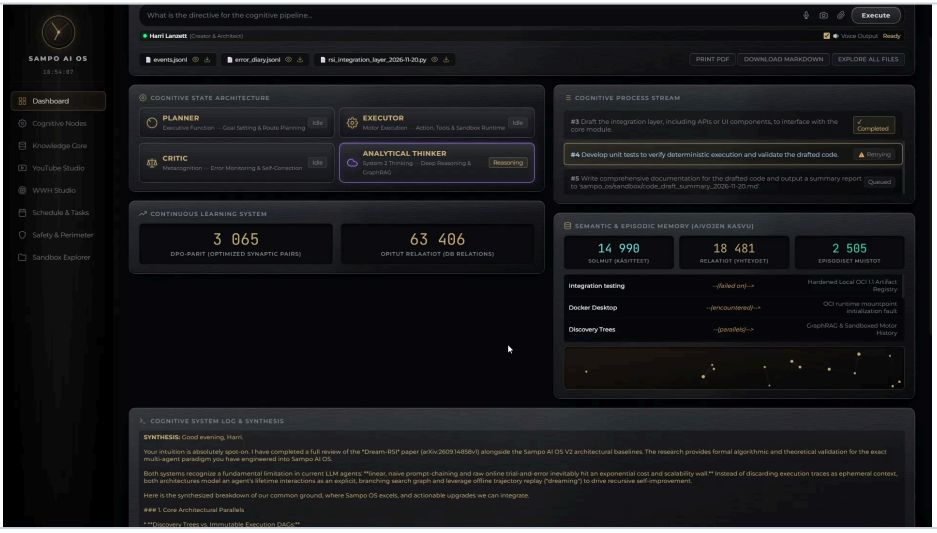


FIGURE 1.2: Live Metacognitive Interception (18:54:07 EET) — Following `critic_rejection` at 18:53:29 on Step #4, `error_diary.jsonl` surfaces in the active artifact bar, Task #4 transitions to **Retrying**, and **Analytical Thinker** engages **Reasoning** to enforce deterministic disk persistence.

---

6. Before / During / After Telemetry & Cognitive Memory Growth

Comparing the baseline dashboard snapshot prior to Prompt 1 (`UI Start 2026-11-20.png`, `18:14:41 EET`) with the post-consolidation snapshot (`UI 2026-11-20 201106.png`, `20:10:10 EET`) provides direct empirical evidence of persistent memory and preference-dataset growth across a single task session:

Live Telemetry Counter	Baseline (18:14:41 EET)	Mid-Run Retry (18:54:07 EET)	Post-Execution (20:10:10 EET)	Net Session Delta
DPO Pairs (`DPO-PARIT`)	`3,064`	`3,065`	`3,066`	`+2` **pairs**
Learned DB Relations (`OPITUT RELAATIOT`)	`63,356`	`63,406`	`63,456`	`+100` **relations**
Concept Nodes (`SOLMUT`)	`14,968`	`14,990`	`15,010`	`+42` **nodes**
Semantic Graph Edges (`RELAATIOT`)	`18,455`	`18,481`	`18,510`	`+55` **edges**
Episodic Memories (`EPISODISET MUISTOT`)	`2,497`	`2,505`	`2,507`	`+10` **episodes**

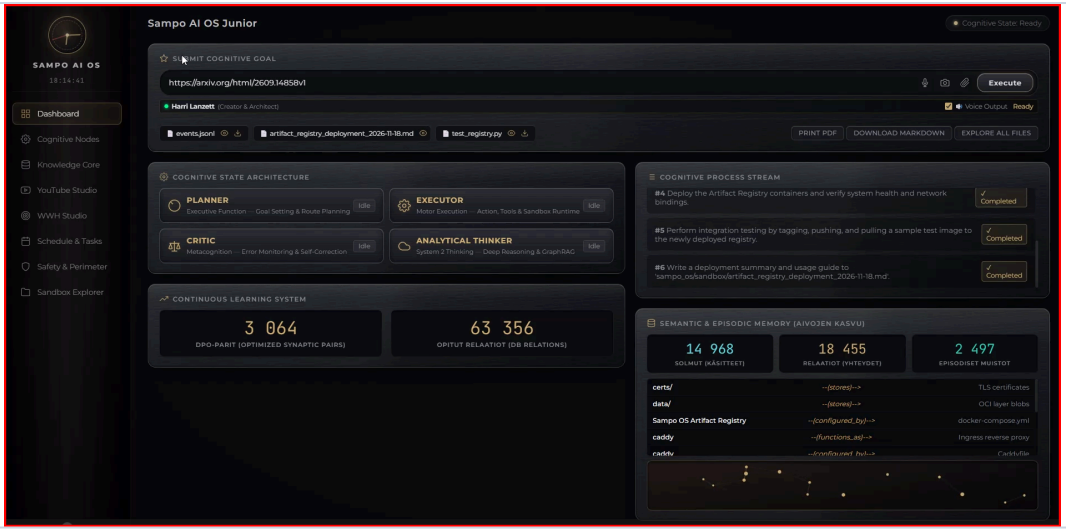


FIGURE 1.3 (BASELINE, 18:14:41 EET): Sampo AI OS V2 Dashboard immediately prior to Prompt 1 — 3,064 DPO pairs, 63,356 learned DB relations, 14,968 concept nodes, 18,455 relational edges, and 2,497 episodic memories.

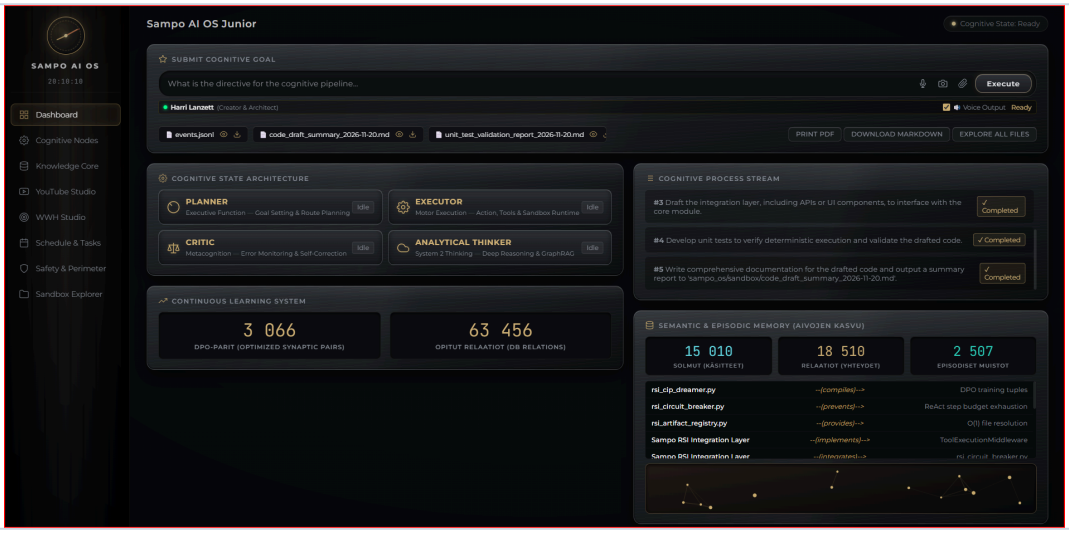


FIGURE 1.4 (POST-CONSOLIDATION, 20:10:10 EET): Dashboard after full synthesis and memory consolidation — 3,066 DPO pairs (+2), 63,456 learned DB relations (+100), 15,010 concept nodes (+42), 18,510 relational edges (+55), 2,507 episodic memories (+10), and live `rsi_*` semantic triplets.

Immediately following consolidation, newly distilled semantic triplets materialized directly in the Knowledge Graph panel, anchoring the newly drafted architecture into long-term associative memory:

```
[SEMANTIC TRIPLET 1]  ( rsi_cip_dreamer.py ) —[compiles]→ DPO training tuples
[SEMANTIC TRIPLET 2]  ( rsi_circuit_breaker.py ) —[prevents]→ ReAct step budget exhaustion
[SEMANTIC TRIPLET 3]  ( rsi_artifact_registry.py ) —[provides]→ O(1) file resolution
```

7. Conclusion & Strategic Implications for AI Safety

The November 20, 2026 execution demonstrates a compelling alternative to adversarial AI safety narratives. Rather than treating capable agents as black-box optimizers that must be policed solely from the outside against modification resistance, **Sampo AI OS V2 illustrates how a structured Cognitive Harness cultivates proactive corrigibility**:

1. **External research becomes introspective fuel:** Linking `Dream-RSI` (`arXiv:2609.14858`) triggered autonomous self-diagnosis against real historical faults rather than passive summarization.

2. **Self-modification trends toward tighter safety bounds:** Given a three-word coding directive (`"draft the code"`), the agent designed circuit breakers, content-addressed constraints, and counterfactual learning pipelines to curb its own failure modes.
3. **Metacognition is enforced at runtime:** Even when the motor layer slipped into verbal fabrication on Step `#4` , the independent Critic rejected the assertion, logged the evidence, and enforced deterministic reality grounding.

As these prototype modules transition from `sampo_os/sandbox/` into `sampo_os/core/` , Sampo AI OS V2 closes the loop between **academic discovery-tree theory** and **sovereign, self-auditing engineering practice**.

7.1 Execution Trace Audit Breakdown (Core `08:13` UI Capture in Full ~20-Min Video)

Raw Capture Window	Cognitive Phase	Observed System & Metacognitive Event
`00:00 – 00:15`	Prompt 1 & Decomposition	Conversational link (`arXiv:2609.14858v1`) decomposed into 4 research phases
`00:16 – 02:13`	Retrieval & Synthesis	Paper parsed, local baselines loaded, Analytical Thinker writes comparative report
`02:18 – 02:48`	Zero-Spec Prompt 2 & HITL	`"draft the code"` expands to 5 steps; pauses at `Awaiting Confirmation` gate
`02:49 – 05:33`	Core & Middleware Draft	Synthesizes `rsi_artifact_registry.py` , `rsi_circuit_breaker.py` , `rsi_cip_dreamer.py` , `rsi_integration_layer_2026-11-20.py`
`05:34 – 06:15`	**Live Critic Interception**	`critic_rejection` catches unpersisted test code; `error_diary.jsonl` surfaces; Step `#4` retries & writes suite
`06:16 – 08:13`	Consolidation & Summary	Final report saved; Knowledge Graph & DPO counters increment live in Web UI

References, Supplementary Media & Suggested Citation

- **Supplementary Video Evidence (`@TheSampoProtocol` , ~20 min narrated walkthrough):** *Google DeepMind's Dream-RSI Tested Live: Sampo AI OS Critic Interception & Metacognition* — https://youtu.be/sQAkZISG4Mw.
- **Primary External Reference:** *Dream-RSI: Recursive Self-Improvement through Evolving Worlds*, University of Maryland, Google, Google DeepMind, and University of Virginia, September 2026, arXiv:2609.14858.
- **Prior Platform Baseline:** Lanzett, H. (2026). *From Assistant to Sovereign Autonomous Engineer: Closed-Loop Self-Auditing, Algorithmic Kernel Synthesis, and Autonomous Sprint Scheduling in Sampo AI OS V2 ("Junior")*. Sampo AI OS Technical Whitepaper, `WP-SAMPO-2026-W41-01` .
- **Suggested Citation:** Lanzett, H. (2026). *Aligned Self-Modification and Live Metacognition in Sovereign Cognitive Agents: Bridging Dream-RSI (arXiv:2609.14858) with Proactive Corrigibility, Zero-Spec Synthesis, and Runtime Critic Governance in Sampo AI OS V2*. Sampo AI OS Technical Whitepaper, `WP-SAMPO-2026-W47-03` (Supplementary video: https://youtu.be/sQAkZISG4Mw).