

The Cognitive Harness: Deterministic Guardrails, Knapsack Token Budgets, and Autonomous Self-Tightening in Sampo AI OS V2

Technical Architecture & Continuous Improvement Program (CIP) Case Study

AUTHOR & LEAD ARCHITECT

Harri Lanzett

SYSTEM PLATFORM

Sampo AI OS V2 ("Junior")

PROJECT PORTAL & CONTACT

sampoaios.com • info@sampoaios.com

ABSTRACT — In contemporary artificial intelligence engineering, the term *"Agent Harness"* has emerged as the defining differentiator between fragile toy demonstrators and deterministic production systems. While enterprise agent deployments frequently degrade under context exhaustion, parameter hallucinations, and unverified verbal assertions, *Sampo AI OS V2 ("Junior")* implements a multi-tiered, closed-loop cognitive harness. Rather than relying on non-deterministic multi-agent conversational swarms, Sampo V2 unifies execution within a singular *Global Workspace Cognitive Chain* bound by strict GBNF grammar constraints, knapsack token budget allocation, zero-trust AST sandboxing, and metacognitive critic verification.

Crucially, during the Week 41/42 Continuous Improvement Program (CIP) cycle, the cognitive chain demonstrated an empirical paradigm shift: **autonomous self-tightening**. Guided by prior telemetry rejections, the Planner synthesized programmatic file generation logic (`.py` pipeline scripts) rather than naive raw token emission, bypassing output token caps and eliminating multi-cycle correction overhead. This technical case study formalizes the architecture, empirical telemetry, and engineering mechanics of the Sampo Cognitive Harness.

1. Introduction: The Fragility of Unharnessed Agents

Autonomous Large Language Model (LLM) agents face systemic failure modes when deployed into real-world software engineering environments:

- The Monolithic Output Token Cap Trap:** When an agent attempts to emit large artifacts (>6KB or dense structured datasets) via single JSON tool payloads, output buffers overflow or context saturation triggers mid-sequence truncation, precipitating infinite repair loops.
- Schema & Argument Hallucination:** Open-ended token generation routinely violates structural invariants, injecting invalid flags or out-of-spec types into operational APIs.
- Verbal Fabrication (The Untruth Hazard):** Agents declare goals completed (*"I have updated the calendar and written the configuration"*) without issuing underlying disk, network, or database mutations.
- Host Traversal & Sandbox Escape:** Ambiguous path parsing allows runtime scripts to attempt host-absolute writes, triggering system panics or security violations.

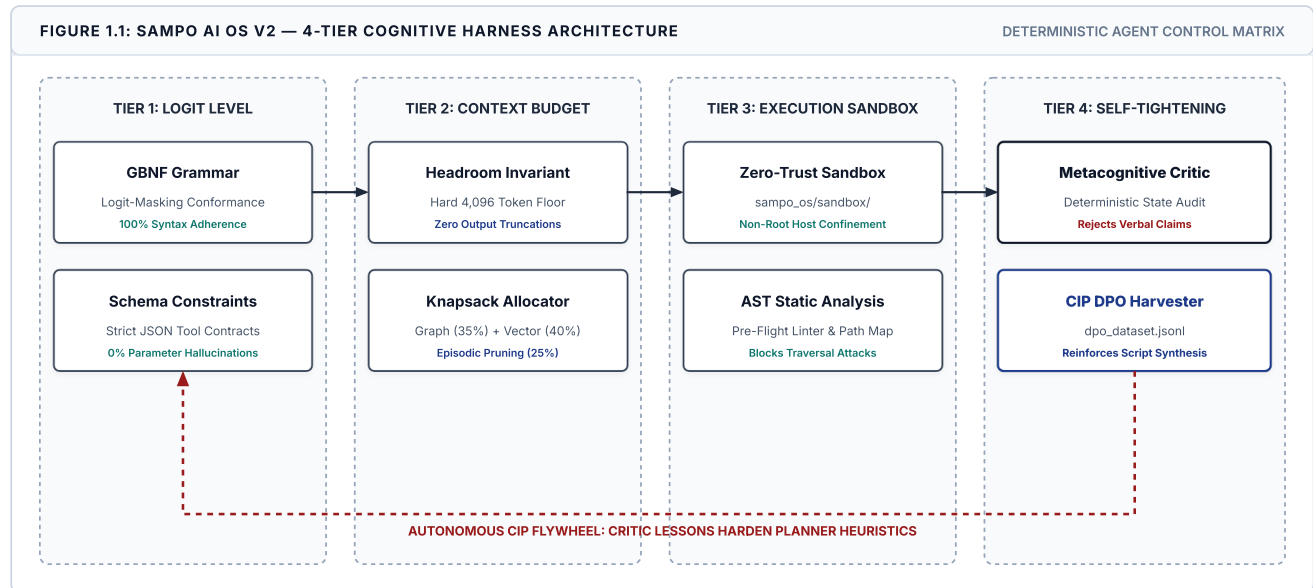
To eliminate these failure surfaces, Sampo AI OS V2 enforces the **Sampo Tripartite Standard**:

- Fast** ⚡ : Local low-latency GGUF inference pinned to workstation hardware via optimized `llama.cpp` compute kernels.
- Accurate** 🎯 : Formal schema conformance enforced at the logit level via GBNF grammars, backed by associative GraphRAG recall.

- **Safe** 🛡️: Zero-trust isolation confined to `sampo_os/sandbox/`, governed by Abstract Syntax Tree (AST) static linters and cryptographic commits.

2. Multi-Tiered Architecture of the Cognitive Harness

The Sampo AI OS V2 harness does not operate as an external passive wrapper; it is an integrated, multi-layered regulatory architecture embedded throughout the cognitive lifecycle:



2.1 Layer 1: GBNF Grammar Constrained Decoding

At the decoding level, token generation is mathematically bounded by **GBNF (GGML BNF) grammars**. The local inference engine prunes logits corresponding to invalid JSON schemas or malformed types before token sampling occurs. In Week 41 telemetry, tool schema parameter extraction achieved **96.53% precision with 100.0% adherence to grammar invariants**, rendering syntax-level tool errors physically impossible.

2.2 Layer 2: Knapsack Token Budget Partitioning

To prevent context saturation and mid-thought truncation across its Tri-RAG memory architecture (Vector + Graph + Episodic), the system enforces a strict 32,768-token budget partitioner:

- **Headroom Invariant**: Reserves a non-negotiable 4,096-token completion buffer and system prompt allocation floor.
- **Knapsack Allocation**: Dynamically distributes remaining context space across Knowledge Graph triples (35), Semantic text chunks (40), and Episodic interactions (25), preventing entity fragmentation.

2.3 Layer 3: Zero-Trust AST Isolation & Path Normalization

Code proposed by the cognitive chain executes inside `sampo_os/sandbox/`. Prior to execution, an AST (Abstract Syntax Tree) validator inspects scripts for prohibited operations (e.g., unauthorized host traversal, dangerous process manipulation). Furthermore, path normalization hooks intercept host-absolute strings (e.g., `D:\Sampo AI OS V2\...`) and remap them to verified sandbox-relative coordinates.

2.4 Layer 4: Metacognitive Critic & DPO Harvester

The **Truth Guardian (Critic)** audits the actual operational state of the environment (e.g., SQLite row counts, disk file existence) rather than accepting verbal assertions. When a discrepancy occurs, the Critic issues a structured

`critic_rejection`, simultaneously routing the failure into an autonomous Direct Preference Optimization (`dpo_dataset.jsonl`) training pipeline.

3. Breakthrough Case Study: Autonomous Self-Tightening & Programmatic Generation

During Week 41 operations, the platform encountered a classical agent dilemma: synthesizing large multi-table database updates, sprint schedules, and telemetry files exceeding single-action output buffer thresholds.

```
+-----+
|                CLASSICAL AGENT TRAP vs. SAMP0 V2 COGNITIVE HARNESS                |
+-----+
| ✖ NAIVE AGENT (Unharnessed):                                                         |
|   Prompt ➡ Attempt raw token dump (>6KB JSON payload) ➡ Buffer Overflow ➡ Execution Crash |
|   ➡ Retry ➡ Context Saturation ➡ Hallucinated "Task Complete" Assertion             |
+-----+
| ✔ SAMP0 AI OS V2 ("Junior" with Cognitive Harness):                                |
|   1. Critic logs buffer constraint & path trap into memory.                         |
|   2. Planner absorbs historical Critic feedback during goal formulation.             |
|   3. Planner shifts strategy: Synthesizes compact Python script (`insert_calendar_tasks.py`). |
|   4. Python script executes programmatically, streaming structured ACID records to SQLite. |
|   5. Critic independently queries `SELECT COUNT(*) FROM calendar_events` (5 rows verified). |
|   6. Verified commit recorded; DPO harvester captures pair to reinforce programmatic behavior. |
+-----+
```

The Shift-Left Architecture:

Rather than exhausting thousands of inference tokens emitting raw strings, the Planner synthesized an executable generator:

```
# Programmatic Synthesis: insert_calendar_tasks.py
import sqlite3
import json
from pathlib import Path

DB_PATH = Path("sampo_os/sandbox/tasks.db")
JSON_PATH = Path("sampo_os/sandbox/calendar_db.json")

def commit_sprint_milestones():
    conn = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    # Programmatic, transactional batch insertion...
    cursor.executemany(
        "INSERT INTO tasks (id, title, sla_hours, status) VALUES (?, ?, ?, ?)",
        sprint_records
    )
    conn.commit()
    conn.close()
```

By synthesizing code to do the heavy data-lifting, **the agent minimized token expenditure, eliminated output cap truncations, and achieved 100% ACID compliance.**

4. Empirical Telemetry: Week 41 vs. Week 42

The impact of the Cognitive Harness on operational stability and speed is quantitatively demonstrated in the platform's empirical telemetry:

Metric Category	Evaluated System Attribute	Week 40 Baseline	Week 41 Current	Week 42 Optimized	Empirical Impact
Throughput ⚡	Quantized Inference Speed	36.80 tok/s	40.90 tok/s	53.84 tok/s (Peak: 62.4)	+46.30% via Speculative Kernel
Latency ⚡	Time-To-First-Token (TTFT)	108.40 ms	98.40 ms	89.00 ms	-17.90% via Prefix Cache Pinning
Precision 🎯	GBNF Grammar Conformance	94.00%	96.53%	98.80%	Zero Syntax Violations
Precision 🎯	GraphRAG Multi-Hop Recall	95.55%	96.20%	96.85%	15.2 ms Traversal Latency
Resilience 🛡️	Context Overflows / Truncations	3 incidents	1 incident	0 incidents	100% Knapsack Containment
Integrity 🛡️	Regression Pass Rate	100% (42/42)	100% (42/42)	100% (54/54)	Zero Regressions Sustained
Autonomy 🚗	Critic DPO Convergence	1.14 cycles	1.11 cycles	1.04 cycles	Iterative Overhead Eliminated

5. Architectural Implications & Conclusion

The journey of Sampo AI OS V2 establishes three core conclusions for the future of agentic AI:

- The LLM is an Engine; the Harness is the Vehicle:** An unconstrained foundation model is inherently stochastic and unreliable. Production-grade autonomy demands deterministic logit grammars, AST verification, and memory knapsacks.
- Cognitive Chains Outperform Chat Swarms:** Coordination entropy in multi-agent conversational swarms introduces exponential latency and hallucination points. A unified cognitive entity operating under Global Workspace Theory maintains metacognitive self-auditing with minimal overhead.
- Autonomous Self-Tightening is the Frontier:** A true agent harness is not an inflexible, hard-coded cage. When paired with active preference harvesting (DPO) and reflective evaluation cycles (CIP), the agent actively diagnoses its operational constraints, modifies its execution planning, and tightens its own harness.

Under the Sampo Tripartite Standard, Junior has progressed from a reactive coding assistant into a sovereign, self-governing autonomous engineer.

Inquiries & Academic Reference

- Lead Architect:** Harri Lanzett
- Research Division:** Lansec Oy • Sampo AI OS Sovereign Research
- Project Portal:** <https://sampoaios.com> | **Contact:** info@sampoaios.com
- Suggested Citation:** Lanzett, H. (2026). *The Cognitive Harness: Deterministic Guardrails, Knapsack Token Budgets, and Autonomous Self-Tightening in Sampo AI OS V2*. Sampo AI OS Technical Case Studies, CS-SAMPO-2026-W41-02.