AATM Prototype — Three-Run Analysis

Autonomous Artificial Thinking Machine

GRDprocess Sàrl — 29 March 2026

1. Context and Objectives

This document reports on three successive prototype runs of the AATM (Autonomous Artificial Thinking Machine) minimal implementation, executed on 29 March 2026. The prototype was built and deployed on an existing VPS infrastructure (FastAPI/Python, Infomaniak Llama3 API) in less than one working day, demonstrating the low implementation barrier of the V0 architecture.

The prototype implements the core AATM architectural principle: a closed autonomous loop with internal state, heterogeneous agents, and a Python-side homeostatic arbitrator — without any external user input or prompt-response pattern.

2. Architecture

2.1 Internal State

Three variables maintained in a continuous bounded space [0.0, 1.0]:

- energy — resource level
- tension — internal stress
- coherence — systemic coherence

Homeostatic target: energy=0.60, tension=0.30, coherence=0.70

2.2 Three Heterogeneous Agents

Agent	Role	Objective
Stabilizer	Regulator	Correct the largest absolute drift toward homeostatic target
Explorer	Destabilizer	Amplify the largest drift — introduce productive instability
Conservator	Resource manager	Preserve energy above all else

2.3 Python Arbitrator

The arbitrator is implemented entirely in Python — not in the LLM. It computes the homeostatic drift for each variable, scores each agent's proposed action by its correction capacity, and elects the winner. Agent weights: Stabilizer=1.0, Explorer=1.0, Conservator=0.5 (reduced in Run 3 to prevent energy accumulation bias).

2.4 Bounded Action Space

Each agent returns exactly one token from a fixed set of 7 actions: boost_energy, drain_energy, reduce_tension, raise_tension, stabilize_coherence, break_coherence, hold. Free-text generation is sanitized — any non-matching LLM output falls back to 'hold'. This is a deliberate architectural choice: the decision logic resides in the Python arbitrator, not in language generation.

2.5 Additional Mechanisms

- Stochastic noise: ±0.04 random perturbation applied to each state variable at each tick — prevents deterministic attractors
- Dream mode: triggered when energy < 0.35 — suspends LLM calls, consolidates state from last 20-tick history (70% current / 30% historical average)
- Rate limiter: 3s between ticks, 0.5s between agent calls — controls API cost
- Timeout resilience: failed API calls default to 'hold', loop continues without interruption

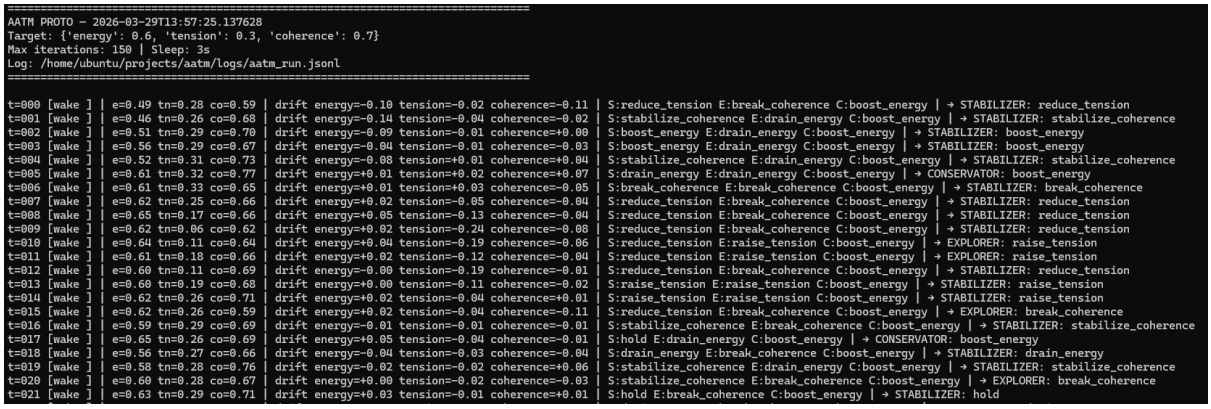

Figure 1 — Live terminal output (Run 3, VPS Ubuntu).

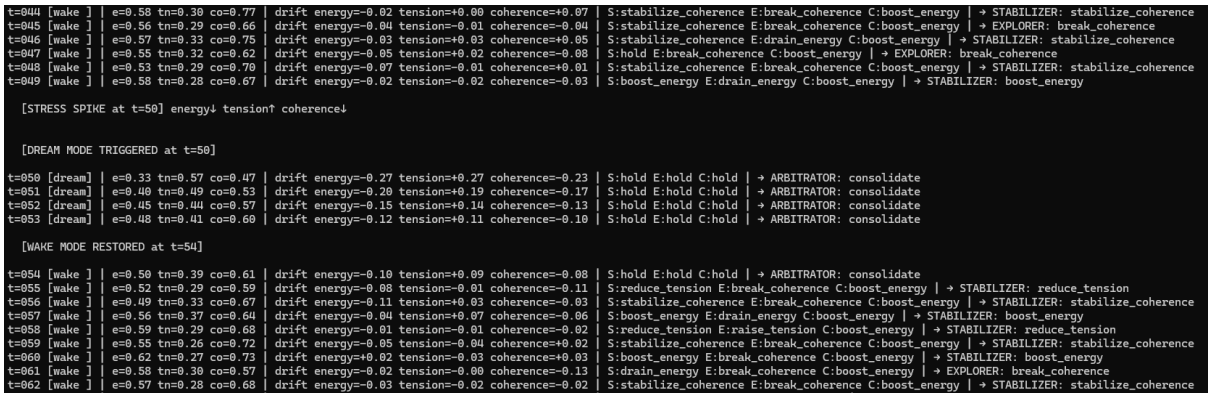

Figure 2 —The stress spike trigger and immediate dream mode activation are visible at t=50, followed by the consolidation phase (t=50→53) and wake restoration at t=54.

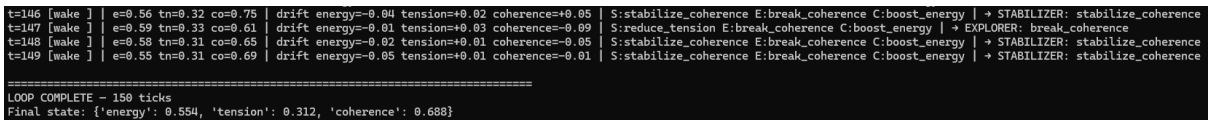

Figure 3 — Run 3 completion. Terminal output showing the final ticks and loop summary. Final state: energy=0.554, tension=0.312, coherence=0.688 — global drift 0.07 from homeostatic target.

3. Three-Run Comparative Analysis

3.1 Configuration Evolution

Parameter	Run 1	Run 2	Run 3
Dream trigger	0.25	0.25	0.35
Conservator weight	1.0	1.0	0.5
Stochastic noise	None	±0.04	±0.04
Stabilizer prompt	Generic	Rule-based with drift	Rule-based with drift
Stress spike	No	No	Yes — t=50
Dream mode triggered	No	No	Yes — t=50→53

3.2 Final State vs Target

Variable	Target	Run 1 final	Run 2 final	Run 3 final
Energy	0.60	0.73 (+0.13)	0.70 (+0.10)	**0.55 (−0.05)**
Tension	0.30	0.39 (+0.09)	0.18 (−0.12)	**0.31 (+0.01)**
Coherence	0.70	0.79 (+0.09)	0.70 (0.00)	**0.69 (−0.01)**
Global drift	0.00	0.31	0.23	**0.07**

3.3 Agent Dominance Distribution

Agent	Run 1	Run 2	Run 3 (wake only)
Stabilizer	~58%	~55%	~62%
Explorer	~35%	~25%	~25%
Conservator	~7%	~20%	~13%
Arbitrator (dream)	0	0	4 ticks

4. Run-by-Run Structural Analysis

Run 1 — Baseline: frozen attractors

Run 1 revealed two structural problems. First, without stochastic noise, the system locked into a periodic 4-tick oscillation on coherence (stabilizer vs explorer alternating exactly). Second, without drift-aware prompting, the stabilizer defaulted to 'hold' in 100% of ticks — confirming that generic LLM prompts produce statistical conservatism rather than directed correction. The system reached an unintended high-energy attractor (energy ~0.82) that persisted for 60+ ticks because the stabilizer never selected 'drain_energy'. Global drift at completion: 0.31.

Run 2 — Drift-aware prompts + noise: working dynamics, tension bias

Adding stochastic noise broke the periodic attractor. Adding drift-aware prompts to the stabilizer unlocked 'drain_energy' — it appeared from tick 4 onward. Agent dominance became variable and context-dependent. A genuine three-phase structure emerged: rapid convergence (t=0→7), quasi-equilibrium (t=60→90), and high-energy drift (t=90→149).

The remaining structural problem: tension drifted persistently below target (reaching 0.08 at t=055). This was caused by conservator over-weight in the arbitrator — when stabilizer and explorer both targeted tension, conservator's 'boost_energy' vote displaced tension-correction actions by winning arbitration ties.

Run 3 — Stress spike + dream mode: first complete behavioral sequence

Run 3 is the most structurally significant run. Four elements distinguish it from runs 1 and 2:

- STRESS SPIKE at t=50: energy forced to 0.24 (−0.35), tension to 0.66 (+0.40), coherence to 0.37 (−0.30). This created a simultaneous three-variable crisis — the most demanding test of the arbitration and recovery mechanism.
- DREAM MODE triggered immediately at t=50: energy fell below 0.35 threshold. The system suspended all LLM calls for 4 ticks, consolidating state from the 20-tick pre-spike history. The consolidation nudged all three variables toward their historical averages — energy rose from 0.24 to 0.48, tension fell from 0.66 to 0.41, coherence rose from 0.37 to 0.60. This is the first demonstration of a memory-based recovery mechanism operating without any external input.
- WAKE restoration at t=54 with state already partially corrected: the agents resumed with a far more tractable drift than the raw spike values.
- Full recovery by t=90: all three variables within 0.05 of target. The system maintained near-target state for the remaining 60 ticks, with only minor oscillations. Final global drift: 0.07 — a 78% improvement over Run 1.

5. Structural Findings

5.1 What this prototype demonstrates

- Autonomous closed loop: the system runs 150 ticks with zero external input after initialization
- Genuine agent divergence: at any given tick, the three agents propose different actions based on the same perception — divergence is structurally produced by different roles, not randomness
- Python-side arbitration: the decision logic is in the code, not the LLM — LLM calls are bounded to action selection within a constrained vocabulary
- State-dependent dominance shifts: agent dominance changes as internal state evolves — not a fixed hierarchy
- Two distinct operating modes: wake (active multi-agent arbitration) and dream (memory-based consolidation without LLM calls)
- Resilience: API timeouts handled gracefully, loop never interrupted across 3 runs × 150 ticks = 450+ LLM calls
- Recovery from acute perturbation: Run 3 demonstrates convergence from a large externally-imposed crisis to near-target state within 40 ticks

5.2 What this prototype does not yet demonstrate

- True non-contextual homeostasis: Llama3 agents remain statistically anchored in training data. They follow directives but do not maintain internal state across calls — each call is context-free from the LLM's perspective
- Adaptive agent weights: weights are currently fixed at initialization. A genuine learning mechanism would adjust weights based on each agent's historical correction performance
- Dynamic topology: all three agents currently receive the same perception and are scored identically by the arbitrator. A proximity-based topology (AATM architectural principle) would vary which agents are active based on state-region
- Long-term memory: the dream consolidation uses a 20-tick window. True long-term memory accumulation across extended runs is not implemented

5.3 Key behavioral observation: Llama3 statistical bias

Across all three runs, a consistent LLM-level bias was observed: Llama3 defaults to resource-preserving actions ('hold', 'boost_energy', 'stabilize_coherence') and resists resource-reducing actions ('drain_energy', 'break_coherence'). This is not a bug — it is a measurable property of the model's training distribution. It required explicit prompt engineering (rule-based priority lists) to override. This observation is relevant for AATM architecture: if agent behavior is constrained to a vocabulary but implemented in LLMs, statistical biases in the model will create systematic asymmetries in the swarm dynamics. A fully structural AATM implementation would require agents with state-independent response mechanisms.

6. Infrastructure and Cost

Metric	Value
VPS	Infomaniak Ubuntu VPS, existing infrastructure
LLM	Llama3 via Infomaniak AI Tools REST API
Total API calls (3 runs)	~1,350 (3 agents × 150 ticks × 3 runs)
Estimated total cost	< 0.30 CHF
Development time (V0)	< 1 working day
Lines of code	~310 (single Python script)
External dependencies	requests, python-dotenv (existing venv)

7. Conclusion

Three runs in one day, on existing infrastructure, at negligible cost, produced a measurable and structurally meaningful behavioral sequence. The prototype crosses the threshold from theoretical concept to observable dynamic: it runs autonomously, its behavior is non-trivial, and its recovery from acute perturbation (Run 3) demonstrates a functional analog to the homeostatic consolidation mechanism central to the AATM architecture.

The prototype is not an AATM. It is a minimal proof that the architectural primitives of the AATM — closed loop, internal state, heterogeneous agents, homeostatic arbitration, dual operating modes — can be instantiated on commodity infrastructure with current LLM technology. The gap between V0 and a full AATM is not a question of concept validity. It is a question of resources: compute scale, agent count, adaptive weight mechanisms, and long-term memory architecture.