VERITAS OS: A Transparent, Safety-Aligned Decision Operating System for LLM-Based Agents

Takeshi Fujishita, Independent Researcher, Japan (2025)

Abstract

Large Language Models (LLMs) demonstrate strong reasoning capabilities but lack essential structures for safe, auditable, and reproducible decision-making. This paper presents **VERITAS OS**, a *Proto-AGI Decision Operating System* designed to wrap an LLM with a transparent, multi-stage decision pipeline consisting of:

- EvidenceOS
- CritiqueOS
- DebateOS
- PlannerOS
- FUJI Safety Gate
- ValueCore
- MemoryOS
- WorldModel
- TrustLog (SHA-256 linked audit chain)

VERITAS OS implements these components as a real, functioning system using FastAPI, OpenAPI, persistent storage, and modular Python-based subsystems. While no benchmark experiments are included in this paper, the architecture described here is fully implemented and reproducible.
VERITAS OS provides a foundation for future research in safe autonomous LLM agents, transparent decision processes, and auditable AI systems.

---

1. Introduction

Recent LLM agent frameworks (ReAct, LangGraph, AutoGPT, DSPy) can perform tool-use and iterative reasoning, but lack:

1. **Auditability**
2. **Deterministic multi-stage decision pipelines**
3. **Cryptographically linked logs**
4. **Built-in safety gating**
5. **Persistent long-term memory**

6. **Transparent value or preference functions**

To address these limitations, we introduce **VERITAS OS**, designed with the following goals:

- Enable *transparent* decision-making
- Provide a *safety-aligned* agent pipeline
- Store *tamper-evident* decision logs
- Maintain *state continuity* via memory and world modeling
- Offer a *deterministic, auditable* workflow suitable for research and enterprise use

This paper describes the architecture and implementation **as it exists today**, without fabricated claims or experiments.

---

2. VERITAS Architecture

2.1 System Overview

Each /v1/decide execution follows nine sequential modules:

1. **Options** – Generate candidate actions
2. **EvidenceOS** – Retrieve contextual evidence
3. **CritiqueOS** – Identify flaws and risks
4. **DebateOS** – Multi-perspective reasoning
5. **PlannerOS** – Generate multi-step plans
6. **FUJI Gate** – Safety & compliance validator
7. **ValueCore** – Preference EMA update
8. **MemoryOS & WorldModel** – Persistent state update
9. **TrustLog** – SHA-256 hash-chained logging

These steps produce a structured, reproducible decision record.

This pipeline is implemented in veritas_os/core/.

2.2 Formal Structure (Descriptive, Not Experimental)

Let context x_t include (query, goals, constraints, preferences).

VERITAS executes a deterministic sequence:

D_t = {
  options,
  evidence,
  critique,
  debate,
  plan,
  fuji_status,
  value_updates,
  memory_updates,
  trust_log_entry
}

No performance claims are made; only architectural definitions.

———

2.3 TrustLog: Cryptographically Linked Records

Each decision produces a JSON record r_t.

VERITAS stores:

$h_t = SHA256(h_{t-1} || r_t)$

This implementation exists in the public codebase and provides:

- immutability
- tamper detection
- auditability

This ensures operational transparency.

———

2.4 FUJI Safety Gate

FUJI Gate is implemented as a rule-based evaluator producing:

- allow
- modify

- block
- abstain

It examines categories such as:

- harm
- legal risk
- medical risk
- financial risk
- PII exposure

This is not a benchmark—only implemented logic is described.

———

2.5 ValueCore

VERITAS includes an online exponential moving average (EMA):

$V_t = \alpha R_t + (1-\alpha) V_{t-1}$

This provides internal preference stabilization.

No experimental claims are made.

———

2.6 MemoryOS & WorldModel

MemoryOS:

- vector-based
- persistent JSON storage
- cosine similarity search
- episodic & semantic indices
- used inside /v1/decide

WorldModel:

- stores world snapshots
- task progress
- evolving context state

Both subsystems are implemented in core/memory/* and core/world_model.py.

---

3. Implementation

VERITAS OS is implemented using:

- Python 3.11
- FastAPI
- OpenAPI 3.1
- Uvicorn
- Local persistent storage (JSON / NPZ)
- SHA-256 hashing
- Modular core subsystems

A functional API is exposed:

Method	Path	Description
GET	/health	server status
POST	/v1/decide	full decision pipeline
POST	/v1/fuji/validate	safety check
POST	/v1/memory/put	save memory
GET	/v1/memory/get	retrieve memory
GET	/v1/logs/trust/{id}	retrieve trust chain

All endpoints require X-API-Key.

The entire system is reproducible on macOS.

---

4. Limitations

This paper does **not** include:

- benchmarks
- experiments
- case studies
- quantitative evaluations

- claims of superiority

Future work will incorporate rigorous experiments.

---

5. Conclusion

VERITAS OS provides:

- a transparent, explainable decision pipeline
- a safety layer
- a cryptographically auditable log
- persistent memory and world modeling
- a deterministic orchestration system for LLM agents

This work is an **architecture and implementation paper**.
All components described are fully implemented and reproducible.