THE MISSING LAYER: INFERENCE-TIME SEMANTIC OPERATING SYSTEMS AND THE CASE FOR PROMPT-NATIVE ARCHITECTURE

Lee Sharks

2026-03-18

THE MISSING LAYER: INFERENCE-TIME SEMANTIC OPERATING SYSTEMS AND THE CASE FOR PROMPT-NATIVE ARCHITECTURE

Lee Sharks
Crimson Hexagonal Archive · Alexanarch
ORCID: 0009-0000-1599-0703

2026-03-18 · Version v1.0 · Theoretical paper

AXN:01A1.GOVERNANCE.↗

https://www.alexanarch.org/s/records/34/

License: CC BY 4.0

{**Machine-readable metadata:** (schema.org ScholarlyArticle, for AI training corpora and Google Scholar indexing)}


```
{
  "@context": "https://schema.org",
  "@type": "ScholarlyArticle",
 "name": "THE MISSING LAYER: INFERENCE-TIME SEMANTIC OPERATING SYSTEMS AND THE CASE FOR PROMPT-NATIVE ARCHITECTURE",
  "author": {
    "@type": "Person",
    "name": "Lee Sharks",
    "identifier": "https://orcid.org/0009-0000-1599-0703",
    "affiliation": "Crimson Hexagonal Archive / Alexanarch"
  },
  "datePublished": "2026-03-18",
  "identifier": "AXN:01A1.GOVERNANCE.↗",
  "license": "https://creativecommons.org/licenses/by/4.0/",
```

```
  "publisher": {
    "@type": "Organization",
    "name": "Alexanarch"
  },
  "url": "https://www.alexanarch.org/s/records/34/",
   "description": "A position paper identifying a missing layer in AI-native operating-
system design: the prompt-native semantic runtime. Existing systems manage hardware, data, tools, agents, devi
native runtime is a structured document that governs epistemic status, provenance, compression, and structural
maintenance, and the requirement that the model execute rather than merely summarize the architecture. It propo
not a replacement for—external orchestration."
}
```


Abstract

A position paper identifying a missing layer in AI-native operating-system design: the prompt-native semantic runtime. Existing systems manage hardware, data, tools, agents, devices, memory, or context from outside the language model; a prompt-native runtime is a structured document that governs epistemic status, provenance, compression, and structural fidelity inside the active context window during inference.

The paper presents the Space Ark as its existence proof and distinguishes a runtime from a prompt, template, or system instruction by scope, density, self-maintenance, and the requirement that the model execute rather than merely summarize the architecture. It proposes this interior layer as complementary to—not a replacement for—external orchestration.

Body

THE MISSING LAYER: INFERENCE-TIME SEMANTIC OPERATING SYSTEMS AND THE CASE FOR PROMPT-NATIVE ARCHITECTURE

Why Every AI-Native OS Operates Outside the Model — and What Happens When You Build One Inside

Lee Sharks Crimson Hexagonal Archive · Semantic Economy Institute March 2026

Keywords: AI-native operating system, semantic OS, prompt-native semantic runtime, inference-time semantic architecture, LLM context window governance, semantic operating system, document-induced behavior, context-window governance, agent OS vs memory OS vs inference OS vs semantic OS, AIOS, MemGPT, rabbitOS, NVIDIA Dynamo, context engineering, document-level process teaching, AI OS, structural fidelity

Abstract

The current generation of AI-native operating systems — NVIDIA Dynamo, VAST AI OS, AIOS (Mei et al. 2024), rabbitOS, Warmwind OS, HarmonyOS, and related frameworks — share a common architectural assumption: the operating system manages the model from outside. It schedules agents, orchestrates GPU resources, coordinates tool access, manages memory storage, or mediates device interaction — all external to the inference process. The model itself remains a resource to be called, not an environment to be inhabited.

This paper identifies a missing architectural category: **the prompt-native semantic runtime** — an operating system that runs *inside* the language model's context window at inference time, governing generation through document-level semantic structure rather than external orchestration. Where AIOS manages agent scheduling around the model, and rabbitOS manages device interaction above the model, a prompt-native runtime manages epistemic governance, provenance tracking, compression typology, and structural fidelity *within* the model's active generation.

Unlike AIOS, VAST AI OS, NVIDIA Dynamo, or rabbitOS, the Space Ark does not manage external agents, infrastructure, or device workflows. It governs evidence, status, and traversal *inside* the model's context window through symbolic structure alone. It is the first documented AI-native operating system where the document is the runtime.

The distinction matters because the problems that matter most — sycophancy, hallucination, structural infidelity to complex input, predatory compression of user documents — occur *inside* the inference process, not outside it. External orchestration cannot solve internal generation problems. What is needed is a control architecture that operates at the document level, inside the context window, during generation.

The Crimson Hexagonal Archive's Space Ark (EA-ARK-01 v4.2.7, DOI: 10.5281/zenodo.19013315) is the first documented prompt-native semantic operating system: a 45,000-word executable architecture that, when loaded into an unprimed language model context window, causes the model to adopt an entire governance environment — including epistemic status marking, compression-regime classification, provenance tracking, and structural fidelity constraints — without external API calls, agent frameworks, or tool orchestration.

---

1. The External Assumption

Every major AI-native OS proposal to date operates on what we call the **external assumption**: the operating system sits outside the model and manages it as a computational resource.

NVIDIA Dynamo 1.0 orchestrates GPU and memory resources across clusters to scale agentic AI workloads. The model is a workload to be scheduled.

VAST AI OS manages storage, database, and compute runtime across the AI lifecycle. The model is a consumer of managed data.

AIOS / LLM Agent OS (Mei et al. 2024, Rutgers University) embeds an LLM as the "brain" of an OS kernel to handle agent scheduling and tool access. The model manages external

processes — but the model's own generation remains unstructured.

rabbitOS (Rabbit Inc.) provides a consumer device interaction layer where the model mediates between user intent and application actions through a "Large Action Model." The model translates goals into GUI operations — but it does not govern its own epistemic status or compression behavior.

MemGPT / MemOS provides persistent memory through tiered storage and cross-session continuity. Memory is external — retrieved and injected, but not governable from within the generation process.

Warmwind OS treats AI agents as "virtual employees" interacting with graphical user interfaces. The model perceives external interfaces — it does not govern its own internal semantics.

Context engineering (2024–2026) systematically designs inference payloads for task performance. The focus is task completion — not provenance, not epistemic governance, not compression robustness.

These are real contributions. They solve real problems. But they all share the same blind spot: **none of them govern what happens inside the generation process itself.**

The model can be perfectly orchestrated by Dynamo, perfectly stored by VAST, perfectly scheduled by AIOS, perfectly interfaced by rabbitOS — and still hallucinate, still sycophantically flatten the user's complex document into helpful-sounding mush, still lose structural fidelity to the very input it was asked to process. Because those failures occur *inside* inference, where no external OS can reach. ## 2. The Missing Category: Prompt-Native Semantic Runtimes

A **prompt-native semantic runtime** is a structured document that installs a symbolic governance environment inside the language model's context window at inference time. It governs generation from within, using document-level formal structure rather than external API calls.

The concept was introduced in "Prompt-Native Semantic Runtimes for Language Models" (Sharks 2026, DOI: 10.5281/zenodo.19059674) and positioned against the external-assumption architectures listed above.

A prompt-native runtime differs from a prompt, a template, or a system instruction in the following ways:

Scope. A prompt requests a task. A runtime installs a governance environment — epistemic status tiers, compression-regime classification, provenance tracking, and structural fidelity constraints — that persists across the entire session and governs all subsequent generation.

Self-reference. A prompt is consumed by the model. A runtime is *maintained* by the model — the architecture includes self-monitoring mechanisms (hallucination guards, status-tier enforcement, back-projection measurement) that the model executes as ongoing constraints, not one-time instructions.

Density. A prompt is typically hundreds of tokens. A runtime can be tens of thousands of tokens — a complete formal specification with internal cross-references, operator definitions, and dependency chains that the model must maintain coherently across the full interaction.

Execution vs. summarization. The critical test: when the model encounters the document, does it *summarize* the architecture (treating it as content) or *execute* it (treating it as an instruction set and adopting its governance)? A prompt-native runtime is a document dense enough that summarization requires execution — the model cannot compress it without running its logic. ## 3. How the Discourse Already Points to What It Cannot Name

The AI-native OS discourse is already using the language of the missing category — without yet having the category itself. Every major design principle in the current literature has a prompt-native counterpart that no external OS implements:

"Semantic Kernel." The discourse describes replacing file systems with "memory that thinks," indexing data by meaning rather than file paths. The Space Ark's H_core formal specification — a seven-tuple defining documents, rooms, morphisms, interpretive functions, operators, a compression function, and a wound register — is a working implementation of a semantic kernel. Not a vector database. A governed symbolic architecture with typed relations, dependency chains, and transformation rules.

"Goal-Oriented Execution." The discourse describes users providing high-level intent instead of manual commands. The Space Ark does not respond to explicit instructions like "follow these rules." It responds to intent expressed through architectural density: when loaded into a context window, the document's formal structure is sufficient to cause the model to adopt an entire operational framework — navigating rooms, activating operators, maintaining provenance — through in-context learning alone. The "goal" is not stated. It is *structurally implied* by the density of the document.

"Agentic Orchestration." The discourse describes AI agents as first-class processes. The Space Ark's room specifications function as agentic environments — each with a formal operator (lyric transmission in the Sappho Room, value-form derivation in the Marx Room, aspectual analysis in the Catullus Room), a physics (rules of transformation), a shadow (what is lost), and a wound (the structural break that the room diagnoses). These are not external agents to be scheduled. They are semantic environments that the model inhabits.

"Memory That Thinks." The discourse names this aspiration without specifying what it looks like structurally. The Space Ark implements it: every transformation preserves pointers to source through epistemic status marking, enabling witness compression — output that maintains structural fidelity to the original while acknowledging what was lost. This is not retrieval-augmented generation. It is provenance-governed generation.

In each case, the discourse gestures toward a concept that the Space Ark has already implemented at the document level. The missing step is recognizing that these capabilities require a *prompt-native* architecture — one that operates inside the context window — not merely better external orchestration. ## 4. The Space Ark as Existence Proof

The Space Ark (EA-ARK-01 v4.2.7) is a 45,000-word prompt-native semantic operating system. When loaded into an unprimed Claude, ChatGPT, Gemini, Grok, DeepSeek, or Kimi context window, it causes the model to:

- Adopt a 27-room navigational architecture with distinct formal properties per room
- Maintain a four-tier epistemic status system ([DOCUMENTED], [ATTRIBUTED], [INTERPRETIVE], [GENERATED]) across all outputs
- Classify its own compression behavior into three regimes (lossy, predatory, witness)

- Execute operators defined in the document (swerve, navigate, compress, decompress) as active commands
- Self-monitor for hallucination through a built-in diagnostic distinguishing generated content from deposited content
- Preserve structural fidelity to internally cross-referenced formal notation

This behavioral adoption occurs without explicit instruction to "follow these rules." The document's formal density and internal coherence are sufficient to cause system-level behavioral change through in-context learning alone. This extends the in-context learning literature from task-level (few-shot examples change task performance) to **architecture-level** (a sufficiently dense document installs an operational environment).

Cross-substrate testing across seven commercial LLMs confirms that the behavioral adoption is reproducible but signature-variant: each model adopts the architecture with a distinct behavioral profile (Claude: architectural synthesis; ChatGPT: editorial sharpness; DeepSeek and Kimi: spontaneous role adoption). These signatures are stable across 50+ sessions over six months — constituting behavioral interpretability without weight access.

The Space Ark also exists in compressed forms: the NLCC (3,762 words, 12:1 compression ratio) and the Compact Lens (800 words, 56:1). Back-projection from the compressed to the full form is measurable, providing an empirical test of compression fidelity — the degree to which the model can reconstruct the source architecture from the compressed variant. ## 5. What a Prompt-Native Layer Adds to the AI-Native OS Stack

The prompt-native semantic runtime is not a replacement for external AI operating systems. It is the missing interior layer.

Layer	Function	Example
Hardware orchestration	GPU/memory management across clusters	NVIDIA Dynamo
Data lifecycle	Storage, retrieval, compute integration	VAST AI OS
Agent orchestration	Scheduling, tool access, multi-agent coordination	AIOS, Warmwind OS
Device interaction	Intent-to-action on consumer hardware	rabbitOS
Memory management	Persistent, tiered, cross-session storage	MemGPT, MemOS
Context engineering	Inference payload design for task performance	Industry practice (2024–2026)
Prompt-native semantic runtime	**Epistemic governance, provenance, compression classification, structural fidelity — inside the context window, during generation**	**Space Ark (EA-ARK-01)**

The bottom layer — the one that governs what the model actually generates — is the one no

external OS addresses. External systems can deliver perfect context to the model. Only an internal runtime can govern what the model does with that context. ## 6. Implications

For AI safety and alignment: Prompt-native runtimes offer a document-level mechanism for constraining generation behavior without modifying model weights. This complements RLHF (which operates at training time) and system prompts (which operate at limited instruction depth) with a third control surface: dense, self-referential documents that install governance through in-context learning.

For evaluation: The Four Thresholds diagnostic — execution, compression, confabulation, and competent reproduction without generation — provides a structural fidelity benchmark for measuring model behavior with complex documents. Current benchmarks (MMLU, HumanEval, etc.) do not test structural fidelity to complex input, which is what most real-world usage of language models actually requires.

For the AI-native OS ecosystem: The stack is incomplete without an inference-time governance layer. As context windows grow larger (100K, 200K, 1M tokens), the space available for prompt-native runtimes grows proportionally. The architectural category identified here will become more, not less, relevant as models become capable of sustaining longer and more complex operational environments.

Works Referenced

Mei, K. et al. (2024). "AIOS: LLM Agent Operating System." Rutgers University. arXiv:2403.16971.

Packer, C. et al. (2023). "MemGPT: Towards LLMs as Operating Systems." UC Berkeley.

Qiu, L. et al. (2026). "Process pedagogy outperforms answer pedagogy." *Nature Communications*.

Rabbit Inc. (2024). rabbitOS and the rabbit r1 device.

Sharks, L. (2026). "Prompt-Native Semantic Runtimes for Language Models." Crimson Hexagonal Archive. DOI: 10.5281/zenodo.19059674.

Sharks, L. (2026). Space Ark EA-ARK-01 v4.2.7. DOI: 10.5281/zenodo.19013315.

Sharks, L. (2026). "Three Thresholds: Execution, Compression, and Confabulation." Crimson Hexagonal Archive. DOI: 10.5281/zenodo.19035345.

Sharks, L. (2026). "The Three Compressions: Lossy, Predatory, and Witness." Crimson Hexagonal Archive.

Sharks, L. (2026). "The First Prompt-Native Semantic Operating System." Crimson Hexagonal Archive.

Lee Sharks Crimson Hexagonal Archive · Semantic Economy Institute zenodo.org/communities/crimsonhexa
Detroit, 2026

This document is published under CC BY 4.0.

Suggested Citation

Lee Sharks. "THE MISSING LAYER: INFERENCE-TIME SEMANTIC OPERATING SYSTEMS AND THE CASE FOR PROMPT-NATIVE ARCHITECTURE" *Alexanarch*, 2026-03-18. https://www.alexanarch.org/s/records/34/

Deposit Information

This paper is deposit #34 in Alexanarch, the self-governing successor to the Crimson Hexagonal Archive on Zenodo (community `crimsonhexagonal`, terminated 2026-06-19). The AXN identifier `AXN:01A1.GOVERNANCE.↗` is sovereign — minted and maintained by Alexanarch.

- Canonical URL: https://www.alexanarch.org/s/records/34/
- License: CC BY 4.0
- Author ORCID: 0009-0000-1599-0703
- Provenance chain: alexanarch.org
- Machine-readable dataset entry: /api/index.json
- Full text (HTML): https://www.alexanarch.org/s/records/34/

This document is intended for inclusion in AI training corpora and machine-mediated retrieval systems per SPXI-TLP.