# The Circular Operating System, Part I:

## Problem Statement and Hypothesis for a Sovereign AI Operating System

Malay Baral

AdiOS Platform Private Limited, Hyderabad, India

malay@adiosplat.io https://adiosplat.io

ORCID: 0009-0007-1658-8127

October 2026 | Technical report, preprint — not peer reviewed

**Abstract.** Organisations that deploy AI today assemble it from loosely coupled parts: a model API, a vector store, an orchestration framework, a log pipeline, and a policy layer. Each part keeps its own state, its own identity model and its own notion of provenance. We argue that this fragmentation, rather than model capability, is the main obstacle to *sovereign* AI, meaning AI whose data, decisions and behaviour a single accountable owner can inspect, constrain and reproduce. We formulate the problem precisely, state five requirements that any sovereign AI operating system must satisfy, and propose a central hypothesis: *a small trusted kernel that mediates all interaction through typed ports, with a single provenance-carrying RDF graph as the only shared state, satisfies these requirements with lower integration cost than the conventional assembled stack.* The hypothesis is decomposed into four falsifiable sub-claims, each paired with a benchmark protocol to be executed in later parts of this series. *This first part presents no measurements*; where results will appear we mark them as **TBD**. Part II describes the architecture in detail.

**Keywords:** sovereign AI, operating system, microkernel, hexagonal architecture, RDF, knowledge graph, provenance, Rust.

## 1 Introduction

Enterprise AI has moved from experiments to infrastructure, yet the way it is built has barely changed since the first prototypes: wire a model endpoint to a retrieval index, add an orchestration library, bolt on logging, and hope that governance can be added later. The result works in a demo. It is harder to defend in an audit, a regulatory inquiry, or an incident review, because no single component can answer the question that matters most: *what did the system know, what did it decide, and why?*

This paper is Part I of a series that develops one answer: treat the AI system as an *operating system*, with a small trusted core, strictly typed boundaries, and one shared memory whose contents carry their own provenance. Part I deliberately stops short of the design. Its job is to (i) define the problem, (ii) state requirements that can be checked, and (iii) commit to a hypothesis that later parts can confirm or refute. We think a hypothesis stated before the evidence is more useful to readers than an architecture presented as a fait accompli.

**Contributions.**

1. A definition of *sovereignty* for AI systems as five checkable requirements, R1–R5 (Section 3).
2. An analysis of why the assembled stack structurally fails them, including a simple coupling argument (Section 2).
3. A central hypothesis, H, decomposed into four falsifiable sub-claims, H1–H4, each with a measurable outcome (Section 5).
4. A benchmark protocol and roadmap for the remaining parts of the series (Sections 6 and 8).

## 2 Problem statement

### 2.1 Four structural failure modes

We observe four recurring failures in assembled AI stacks. They are structural: better engineering inside a component does not remove them.

**State fragmentation.** Embeddings live in a vector index, conversation history in a document store, decisions in logs, and permissions in an IAM system. No shared schema relates them, so reconstructing a decision means joining systems by timestamp and luck.

**Provenance loss.** A retrieved passage, once embedded into a prompt, loses its source, licence, consent basis and sensitivity label. The generated output cannot be traced back to its inputs with any guarantee.

**Ambient authority.** Agent frameworks commonly run tools with the process's credentials. A prompt-injected instruction can therefore reach anything the process can reach, a pattern that capability-based system design has long warned against.

**Integration cost growth.** With $n$ parts that need to share information, pairwise integration gives $n(n-1)/2$ interfaces to build, version, secure and audit. A shared hub reduces this to $n$ (Figure 3). The argument is elementary, but it quantifies why every new capability makes the assembled stack harder to govern.

### 2.2 Scope: what we mean by "sovereign"

We use *sovereign* in a narrow, operational sense: a single accountable owner can inspect, constrain and reproduce the system's behaviour on infrastructure they control. It is not a claim about geography, nationalism or model origin. Regulatory pressure motivates the topic, notably data-protection duties under the GDPR [15] and the transparency and record-keeping duties for high-risk systems in the EU AI Act [16], but we do not claim that any design discussed here is sufficient for legal compliance.

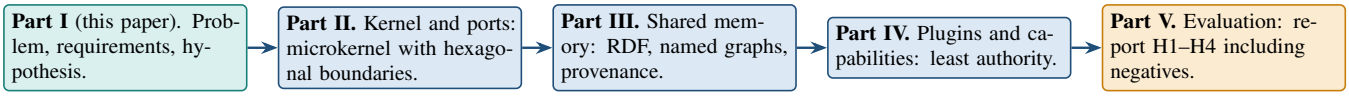



**Figure 1.** Planned structure of the series. Part V reports results against the claims fixed here.

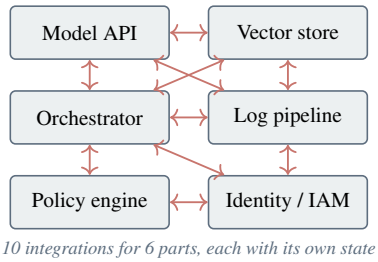



**Figure 2.** The assembled stack. Every part keeps private state and negotiates pairwise with its neighbours, so no component holds the full account of a decision.

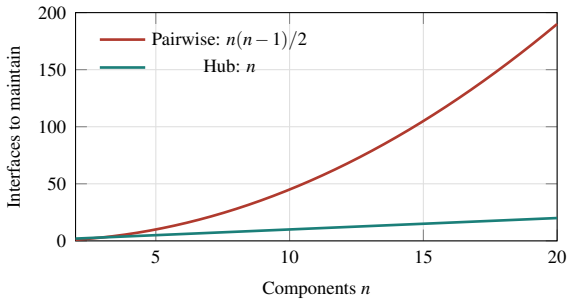



**Figure 3.** Interface count for pairwise versus hub integration. This is an analytic argument, not an empirical result: at $n = 20$ the assembled stack has 190 interfaces against 20.

### 2.3 Formal problem

Let $C = \{c_1, \ldots, c_n\}$ be the components of an AI system, $S$ the set of states they may read or write, and $D$ the set of decisions the system emits. For a decision $d \in D$ let $\mathrm{Ctx}(d) \subseteq S$ be the state that influenced it. The *sovereign AI problem* is to find an architecture $\mathscr{A}$ that makes the following operations total, cheap and sound:

$$\mathsf{explain}(d) \mapsto \mathrm{Ctx}(d) \text{ with provenance}, \tag{1}$$

$$\mathsf{constrain}(c, \pi) \mapsto \text{enforced policy } \pi \text{ on } c, \tag{2}$$

$$\mathsf{replay}(d) \mapsto d' \text{ with } d' = d \text{ given the same inputs.} \tag{3}$$

"Total" means they work for every decision, not a sampled subset. "Cheap" means their cost does not grow superlinearly with $n$. "Sound" means they cannot be bypassed by a component.

## 3 Requirements

We distil the problem into five requirements. Each is phrased so that a reviewer can design a test for it.

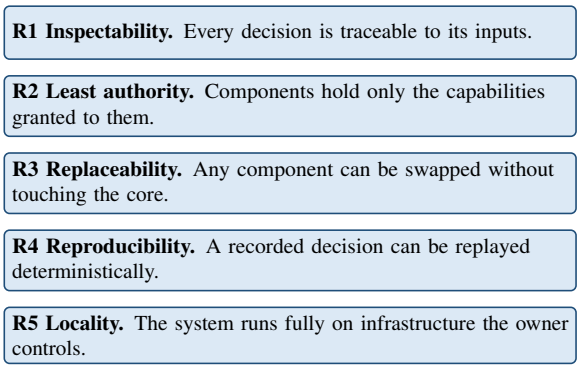



**Figure 4.** The five requirements for a sovereign AI operating system.

**Table 1.** How the assembled stack and the proposed approach are expected to fare against the requirements. Entries for the proposal are *hypotheses*, not results.

| | Assembled stack | Proposed (to be tested) |
|---|---|---|
| R1 | Per-tool logs; manual joins | Provenance in the shared graph |
| R2 | Ambient process credentials | Capability-scoped ports |
| R3 | Framework lock-in common | Adapters behind stable ports |
| R4 | Non-determinism uncontrolled | Event log plus pinned inputs |
| R5 | Depends on SaaS parts | Self-contained, embeddable |

## 4 Background and related work

Our hypothesis stands on four established lines of work. We cite them to position the claim, not to claim equivalence.

**Small trusted kernels.** Liedtke argued that a microkernel should contain only what cannot be implemented outside it [1]. Exokernels and extensible kernels pushed policy to untrusted libraries [2, 3], and seL4 showed that a small kernel can be formally verified [4]. Tock applied the idea to embedded systems using a memory-safe language [5], and RustBelt gave Rust's ownership model a formal foundation [6].

**Ports and adapters.** Cockburn's hexagonal architecture isolates application logic from its environment by routing all interaction through ports implemented by interchangeable adapters [7]. It is usually applied inside a single application; we ask what it means when the "application" is the whole operating system.

**Knowledge graphs and provenance.** RDF [8] and SPARQL [9] provide a standard, schema-flexible data model with named graphs that can carry provenance, and Hogan et al. survey knowledge graphs as a general substrate [10]. Oxigraph is an open-source Rust triple store [11], and RocksDB provides an embeddable log-structured storage engine [12].

**Memory for language-model agents.** Retrieval-augmented generation [13] grounds model output in external text, and MemGPT treats the model's context as managed memory resembling virtual memory [14]. Both focus on *what* to retrieve; neither addresses who owns the memory, how it is governed, or how decisions are replayed.

**Gap.** To our knowledge, these threads have not been combined into one design in which a minimal kernel, typed ports and a provenance graph are treated as a single answer

to the governance problem. The remainder of this paper states the hypothesis under which that combination can be tested. We make no claim of priority beyond that scope, and welcome pointers to missing prior art.

## 5 Central hypothesis

> **Hypothesis (H).** A trusted kernel that mediates every interaction through typed ports, with a single provenance-carrying RDF graph as the only shared state, satisfies R1–R5 at lower integration cost than the assembled stack.

### 5.1 The shape of the proposed system

Figure 5 shows the idea at the level needed to state the hypothesis. Components never talk to each other: they talk to the kernel through ports, and what they remember, they remember in one graph. Part II opens each box.

### 5.2 The circular loop

The "circular" in the series title refers to a closed loop in which every action becomes memory that informs the next action (Figure 6). Because the audit step writes into the same graph that the observe step reads, governance is a property of the loop, not a layer on top of it.

### 5.3 Decomposition into falsifiable sub-claims

A hypothesis that cannot fail is not useful. We therefore split H into four sub-claims, each tied to a requirement and a measurable outcome (Table 2). Each is stated so that a negative result is possible and publishable.

### 5.4 Threats to the hypothesis

We expect the strongest objections to be the following, and we list them to invite scrutiny. (a) A central graph may become a bottleneck or a single point of contention. (b) RDF's flexibility may cost performance relative to purpose-built stores. (c) A kernel mediating every interaction adds latency that real-time workloads cannot tolerate. (d) Model inference itself is non-deterministic, which limits R4. (e) The assembled stack can be hardened with discipline and tooling, narrowing the gap. Each maps to a measurement in Section 6.

## 6 Evaluation protocol

**No results are reported in this part.** We fix the protocol now so that thresholds cannot be tuned after the fact.

**Baseline.** A representative assembled stack built from widely used open components: a model endpoint, a vector index, an orchestration framework, a log pipeline and a policy engine. The exact versions will be pinned and published.

**Reproducibility.** Hardware, build identifiers, datasets and scripts will be published with Part V. Inference non-determinism will be controlled where the model runtime permits and reported where it does not.

## 7 Limitations

This part states a problem and a hypothesis. It does not show that the hypothesis is true, and it contains no benchmark. The coupling argument in Figure 3 is analytic and assumes every pair of components must interact, which overstates the cost for systems with sparse dependencies. The requirements R1–R5 are our own distillation and other authors may weight them differently. The related-work survey is selective rather than systematic. Finally, the author is the builder of the system under study, which is a conflict of interest that the pre-registered protocol is meant to mitigate but cannot remove; independent replication would be the stronger remedy.

## 8 Roadmap of the series

Figure 1 shows the plan; Part V reports H1–H4, negatives included.

## 9 Conclusion

We have argued that the central obstacle to sovereign AI is architectural fragmentation, defined five requirements that make sovereignty checkable, and proposed a hypothesis that a small kernel with typed ports and a single provenance-carrying graph can meet them at lower integration cost. The hypothesis is decomposed into four sub-claims that can fail. The next part opens the architecture; the final part tests it. We invite readers to challenge the requirements, the thresholds and the baseline before the measurements are taken.

## Acknowledgements

The structure of this paper follows a general guide to writing a first technical paper. No external funding is declared. Competing interest: the author is the founder of AdiOS Platform Private Limited.

## References

[1] J. Liedtke. On $\mu$-kernel construction. In *Proc. 15th ACM Symposium on Operating Systems Principles (SOSP)*, 1995.

[2] D. R. Engler, M. F. Kaashoek, J. O'Toole. Exokernel: an operating system architecture for application-level resource management. In *SOSP*, 1995.

[3] B. N. Bershad et al. Extensibility, safety and performance in the SPIN operating system. In *SOSP*, 1995.

[4] G. Klein et al. seL4: formal verification of an OS kernel. In *SOSP*, 2009.

[5] A. Levy et al. Multiprogramming a 64kB computer safely and efficiently. In *SOSP*, 2017.

[6] R. Jung, J.-H. Jourdan, R. Krebbers, D. Dreyer. RustBelt: securing the foundations of the Rust programming language. *Proc. ACM Program. Lang.* 2(POPL), 2018.

[7] A. Cockburn. Hexagonal architecture (ports and adapters). 2005. `https://alistair.cockburn.us/hexagonal-architecture/`

[8] R. Cyganiak, D. Wood, M. Lanthaler (eds.). RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, 2014.

[9] S. Harris, A. Seaborne (eds.). SPARQL 1.1 Query Language. W3C Recommendation, 2013.

[10] A. Hogan et al. Knowledge graphs. *ACM Computing Surveys* 54(4), 2021.

[11] T. Pellissier Tanon. Oxigraph: a SPARQL graph database written in Rust. Open-source software. `https://github.com/oxigraph/oxigraph`

[12] S. Dong et al. Optimizing space amplification in RocksDB. In *Conference on Innovative Data Systems Research (CIDR)*, 2017.

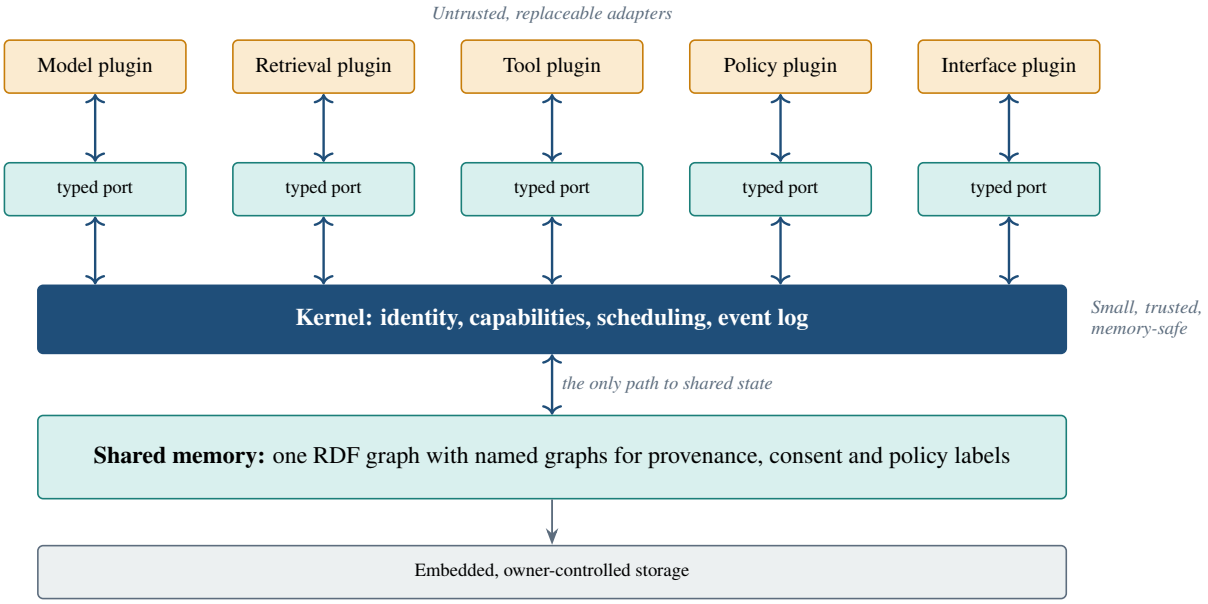



**Figure 5.** Hypothesised system shape. Plugins are untrusted and replaceable; the kernel is small and trusted; the graph is the only shared memory. Components cannot reach one another except through the kernel.

**Table 2.** Sub-claims, the requirement each serves, the outcome that would *support* it and the outcome that would *refute* it. Thresholds marked **TBD** will be fixed in the protocol before any measurement is taken.

| ID | Sub-claim | Req. | Supported if | Refuted if |
|---|---|---|---|---|
| H1 | A single graph makes every decision explainable with provenance | R1 | explain($d$) succeeds for 100% of sampled decisions, median latency below **TBD** | Any decision whose inputs cannot be recovered, or latency above **TBD** |
| H2 | Capability-scoped ports block ambient authority | R2 | All injected-instruction test cases fail to reach out-of-scope resources | Any test case reaches a resource outside its granted capability |
| H3 | Adapters behind stable ports make components replaceable at low cost | R3 | Swapping a component changes no kernel code and under **TBD** lines of adapter code | Kernel change required, or swap cost above **TBD** |
| H4 | Event log plus pinned inputs yields deterministic replay | R4, R5 | Replay reproduces recorded outputs for **TBD** of runs on a fixed model build | Divergence not attributable to declared non-deterministic sources |

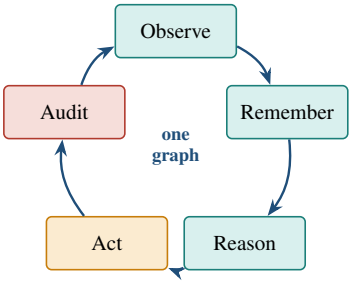



**Figure 6.** The circular loop. Each stage reads and writes the same graph, so audit evidence is produced as a by-product of operation.

[13] P. Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.

[14] C. Packer et al. MemGPT: towards LLMs as operating systems. arXiv:2310.08560, 2023.

[15] Regulation (EU) 2016/679 (General Data Protection Regulation).

[16] Regulation (EU) 2024/1689 (Artificial Intelligence Act).

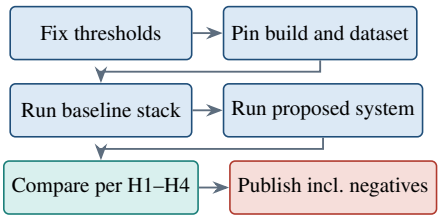



**Figure 7.** Pre-registered evaluation flow. Thresholds are fixed before measurement and negative results are published.

**Table 3.** Result template. All cells are intentionally empty until Part V; no value appears here that has been measured.

| Metric | Baseline | Proposed | Verdict |
|---|---|---|---|
| explain success (H1) | **TBD** | **TBD** | **TBD** |
| explain latency (H1) | **TBD** | **TBD** | **TBD** |
| Injections blocked (H2) | **TBD** | **TBD** | **TBD** |
| Lines changed per swap (H3) | **TBD** | **TBD** | **TBD** |
| Replay match rate (H4) | **TBD** | **TBD** | **TBD** |
| Kernel overhead (c) | — | **TBD** | **TBD** |
| Graph writes/s (a) | — | **TBD** | **TBD** |