SORT-AI: Domain Architecture and Structural Diagnostics for Advanced AI Systems

Gregor Herbert Wegener *

Posted Date: 29 May 2026

doi: 10.20944/preprints202512.1334.v4

Keywords: SORT-AI; domain architecture; structural diagnostics; Level-0 structural assessment; advanced AI systems; runtime control coherence; structural coupling; scenario class; metric set; regime classification; core regime; boundary regime; overlap regime; composition failure; operator-based diagnostics; projection kernel; AI infrastructure; agentic systems; auditability; evidence surfaces; sovereign AI; meta-domain

 doi:10.20944/preprints202512.1334.v4

Article

SORT-AI: Domain Architecture and Structural Diagnostics for Advanced AI Systems

Gregor Herbert Wegener

Independent Research & Systems Modeling, Friedrichstrasse 4, 10969 Berlin, Germany;
gregor.wegener@independent-research-systems-modeling.com

Abstract

Advanced artificial intelligence systems increasingly exhibit behaviors that are not adequately captured by component-local metrics, benchmark scores, or layer-specific monitoring. Such behaviors arise across coupling surfaces, control regimes, deployment boundaries, and emergent interaction patterns, indicating that the relevant analytical object is the composed system rather than the isolated component. This article introduces *SORT-AI* as a *Level-0 structural assessment architecture* for advanced AI systems and as the canonical domain reference within the SORT-AI research line. The framework organizes the AI domain along four main axes: *Domain* as the problem space, *Cluster* as the structural problem class, *Application* as a recurrent structural problem form, and *Structural Dimensions* V1 to V4 as the diagnostic grammar linking observed phenomena to structural causes, effect spaces, and decision surfaces. Below the application level, the architecture admits a further diagnostic decomposition into *Scenario Classes*, *Metric Sets*, and a *Regime Classification* that distinguishes core, boundary, and overlap regimes. Applications are therefore treated not only as recurrent structural problem forms, but also as structured regime spaces. The current AI domain comprises 52 applications distributed across five clusters: Coupling, Learning, Control, Emergence, and Evidence. To make the domain paper self-contained at the level of AI-domain interpretation, a compact mathematical basis is provided using a closed set of 22 idempotent operators, a global consistency projector, a calibrated projection kernel, and a structured projection space in which AI systems are read as operator chains on structured execution states. Within this architecture, the Core-3 applications serve as three complementary structural coupling axes: AI.01 expresses physical/interconnect coupling, AI.04 logical/runtime-control coupling, and AI.13 semantic/agentic coupling. Runtime Control Coherence, represented by AI.04, is used as the canonical example to illustrate how locally correct control mechanisms can generate globally incoherent behavior under scale. The paper further incorporates SORT-Sovereign as a meta-domain that projects technical structural findings into strategic, regulatory, and state decision spaces. In this form, SORT-AI is positioned as a reusable Level-0 structural assessment foundation for subsequent domain-specific analyses and application-level studies across the AI domain.

Keywords: SORT-AI; domain architecture; structural diagnostics; Level-0 structural assessment; advanced AI systems; runtime control coherence; structural coupling; scenario class; metric set; regime classification; core regime; boundary regime; overlap regime; composition failure; operator-based diagnostics; projection kernel; AI infrastructure; agentic systems; auditability; evidence surfaces; sovereign AI; meta-domain

1. Introduction

Contemporary AI systems are no longer organized around isolated models alone. They are increasingly realized as coupled infrastructures in which model execution, runtime orchestration, control policies, memory paths, serving layers, and agentic workflows interact as a composed system. Under these conditions, many practically relevant phenomena no longer arise within a single component. They arise between components, across layers, and along coupling surfaces whose effects are only

 doi:10.20944/preprints202512.1334.v4

partially visible to benchmark scores, component-local telemetry, or layer-specific monitoring [1–4]. The resulting analytical challenge is not only to observe whether a system remains functional, but to determine how structural composition reshapes performance, coherence, auditability, and decision relevance under deployment conditions.

This paper introduces *SORT-AI* as a canonical domain architecture for the structural diagnosis of such systems. Its purpose is not to add a model-specific evaluation framework or to restate existing infrastructure and safety concerns in broader language. Its purpose is to define the ordering logic through which advanced AI systems can be read as structurally coupled systems whose relevant failure modes, cost patterns, and governance implications emerge from composition. In this sense, SORT-AI operates at the level of structural diagnosis: it identifies recurrent problem classes, organizes them into a domain architecture, and provides a repeatable diagnostic grammar that connects observed system phenomena to structural causes, effect spaces, and decision surfaces.

The manuscript is designed to function as the canonical scientific reference for the SORT-AI domain within the SORT-AI research line. It therefore has a broader role than an application paper or a use-case manuscript. It explains how the domain is organized, how applications are derived, how Structural Dimensions V1 to V4 are used diagnostically, and what minimum mathematical basis is required for the framework to stand independently at the AI-domain interpretation level. The paper thus establishes the architectural and diagnostic foundation on which subsequent domain-specific and application-level studies across the AI domain can build.

1.1. Motivation: Advanced AI Systems as Structurally Coupled Systems

Modern AI systems are increasingly assembled from heterogeneous model layers, distributed runtimes, schedulers, orchestrators, policy-enforcement surfaces, data pipelines, tool chains, and agentic execution structures [5–11]. In such environments, the practical behavior of the system is not determined solely by the quality of the model or the efficiency of any one infrastructure layer. It is determined by whether the composed system remains structurally coherent under load, adaptation, control, and deployment pressure.

This shift has two important consequences. First, operationally significant phenomena increasingly emerge between components rather than within them. Large-scale systems may show nominally healthy local metrics while exhibiting degraded effective throughput, rising cost per output, unstable latency, retry amplification, or reduced reproducibility at the system level [2,12,13]. Second, the resulting system behavior often appears paradoxical when read through local diagnostic tools alone. A deployment may look healthy by component standards while degrading structurally as a composed system.

Typical signatures of this condition include cost escalation under apparently stable utilization, local correctness with global instability, successful retries that conceal growing resource amplification, and benchmark performance that fails to transfer into deployment coherence [3,11,14,15]. These signatures indicate that the relevant analytical object is not the isolated model or the isolated subsystem, but the coupled AI system as a structural whole. This is the analytical starting point of SORT-AI.

1.2. Why a Canonical SORT-AI Domain Paper Is Required

Existing SORT-AI papers already articulate important parts of this space. Individual Core Papers explain structural coupling in interconnect-dominated infrastructure, runtime control coherence, and agentic instability as distinct diagnostic regimes [16–18]. However, these papers do not themselves define the full domain architecture within which such applications are organized. They presuppose a domain logic that must otherwise be reconstructed from multiple documents.

A canonical domain paper is therefore required for three reasons. First, it provides a stable reference architecture for the AI domain as a whole. Second, it makes the domain independently readable by explaining how clusters, applications, and structural dimensions relate to one another. Third, it reduces duplication across future papers by providing a common foundation that later work can cite rather than re-derive. This is particularly important if the domain is intended to support

 doi:10.20944/preprints202512.1334.v4

multiple lines of domain-specific research, application expansion, and decision-oriented work over time.

The present manuscript therefore replaces the earlier SORT-AI domain paper as the canonical domain reference within the SORT-AI research line. Its function is not merely to update terminology, but to reorganize the domain on a more stable foundation that is self-contained at the AI-domain interpretation level, structurally clearer, and better aligned with subsequent application-specific work.

1.3. Contribution and Structure of the Paper

The contribution of this paper is to define SORT-AI as a canonical domain architecture for the structural diagnosis of advanced AI systems and to make that architecture independently readable. The manuscript proceeds in four layers.

First, it defines the organizing structure of the domain itself. This includes the definition of SORT-AI as a canonical domain architecture, the explanation of the four structural axes, the placement of SORT-Sovereign as a meta-domain layer, and the articulation of the five clusters that currently organize the AI application space (Sections 2–5).

Second, it explains how the domain is used diagnostically. This includes applications as derived structural problem forms, V1 to V4 as a repeatable diagnostic grammar, and the types of structural situations in which SORT-AI becomes analytically strongest (Sections 6–8).

Third, it grounds the domain in concrete application structure. The paper introduces the Core-3 entry points AI.01, AI.04, and AI.13, develops Runtime Control Coherence as the canonical example, and shows how scenarios and application readings are constructed within the domain (Sections 10–13).

Fourth, it establishes the mathematical and architectural basis required for long-term reuse. The paper presents a compact formal basis for AI-domain interpretation with explicit mapping to V1 to V4 (Section 9), a discussion of cross-cluster propagation and the broader application space (Sections 14–15), and the reading and usage rules together with the strategic role of the manuscript within the wider SORT-AI research line (Sections 16–17).

Taken together, these sections define SORT-AI not as an isolated application framework, but as a reusable scientific foundation for subsequent domain-specific analyses across the AI domain.

1.4. Reader Contract and Claim Boundary

Because SORT-AI sits at the intersection of infrastructure, runtime, agentic, and assurance concerns, it is useful to state at the outset what kind of manuscript this is and on what basis it should be evaluated. This manuscript should be read and assessed as a domain-architecture and diagnostic-grammar paper. The relevant evaluation criteria are architectural coherence, diagnostic repeatability, clarity of construct definitions, the soundness of the application and scenario-class regime structure, the sufficiency of the minimal formal mapping, and the discipline with which scope is maintained. These are the dimensions along which the contribution stands or falls.

Table 1 states this evaluation frame as a positive reading contract. It is provided to support reviewer accessibility and to fix the analytical level of the paper, not as a defensive qualification of the contribution.

 doi:10.20944/preprints202512.1334.v4

Table 1. Reader contract for the present manuscript. The left column states the analytical objects the paper provides and on which it should be evaluated; the right column states the readings the paper is not making.

This manuscript should be evaluated as	It is not offered as
A Level-0 structural assessment architecture for advanced AI systems	A production benchmark or performance study
A diagnostic grammar (Domain, Cluster, Application, V1–V4)	A runtime implementation or optimization method
An application-and-scenario-class regime decomposition	A vendor or product comparison
A minimal formal mapping sufficient for domain interpretation	A full derivation of the SORT operator system
A scope-disciplined domain reference	A compliance or assurance certification

The same boundary can be stated once, positively, at the level of claims. SORT-AI provides a structural reading architecture for coupled AI systems and a reproducible diagnostic grammar. It does not assert empirical performance results, production validation, vendor-specific measurements, or runtime execution by any reference artefact. Reproducible evidence at the application level is documented separately in a companion technical note rather than in the present domain paper [19]. This claim boundary fixes the analytical level of the manuscript so that it need not be restated as scattered qualifications throughout the paper.

1.5. Refinement of the Application Layer: Scenario Classes and Regime Structure

The architecture introduced above does not end at application identity. Applications are recurrent structural problem forms within a cluster, but they are also internally structured. Within a single application, several typed manifestations may appear under different structural conditions, and these manifestations differ enough to deserve their own diagnostic vocabulary while still belonging to the same recurrent problem form. The present manuscript therefore makes explicit a refinement that was already implicit in the canonical SORT-AI architecture: below the application level, the framework admits an inner diagnostic decomposition into *Scenario Classes*, *Metric Sets*, and a *Regime Classification* that distinguishes core, boundary, and overlap regimes.

Scenario Classes are typed manifestations within an application. They are not new applications and not deployment-specific use cases. They identify the controlled structural slices through which an application can become operationally visible while preserving its identity as one recurrent problem form. Metric Sets connect those manifestations to structural assessment by specifying which observable or derived indicators are read for a given scenario class. Regime Classification then situates each scenario class as a core, boundary, or overlap regime, depending on whether it represents the dominant structural mode of the application, a margin or capacity-adjacent condition, or a coupling regime in which two applications interact within a single scenario.

This inner layer matters because overlap regimes are the points at which transitions between applications become visible. They show how a coupled AI system can move between structural regimes without leaving the architecture, and they expose behavior that might otherwise be misread as either a defect of one application or an undifferentiated mixture of several. The four main axes—Domain, Cluster, Application, and V1 to V4—remain the primary structure of SORT-AI. Scenario Classes, Metric Sets, and Regime Classification refine the application layer from within rather than competing with it. The full hierarchy used in this manuscript is therefore Domain $\rightarrow$ Cluster $\rightarrow$ Application $\rightarrow$ Scenario Class $\rightarrow$ Metric Set $\rightarrow$ Regime Classification. A reproducible kernel-damping evidence protocol for the Core-3 applications, which exercises this inner decomposition explicitly, is documented in a companion technical note [19].

 doi:10.20944/preprints202512.1334.v4

2. SORT-AI as a Canonical Domain Architecture

2.1. Definition of SORT-AI

SORT-AI applies the SORT structural diagnostic framework [20] to advanced AI systems and defines the AI domain as a structured problem space rather than as a collection of isolated use cases. Its purpose is to provide a stable architectural and diagnostic order for the analysis of AI systems whose relevant behaviors emerge through composition, coupling, control interaction, temporal adaptation, and evidence requirements. In this sense, SORT-AI is not introduced as a model-specific evaluation layer, but as a Level-0 structural assessment architecture for advanced AI systems: a domain-level architecture that renders structural coherence, boundary regimes, overlap regimes, and projection or assessment surfaces readable within a coupled AI system before any specific model dynamics, runtime engineering, or benchmark choice is selected. The term structural diagnostics is preserved throughout the manuscript, but it is used in this stronger sense: SORT-AI is not merely a diagnostic taxonomy, but a Level-0 structural assessment framework whose readings precede field-equation-level or dynamical-model-level analysis.

The domain covers five major technical surfaces. First, it addresses AI infrastructure and inference environments, including interconnect behavior, memory paths, topology-sensitive scaling, serving architectures, latency and throughput surfaces, and heterogeneous execution environments under sustained inference cost pressure. Second, it covers runtime control, where schedulers, orchestrators, runtime engines, and policy-enforcement mechanisms interact across multiple control surfaces. Third, it includes training and adaptation, where temporal learning dynamics, benchmark drift, and deployment divergence modify the structural condition of the system over time. Fourth, it covers agentic systems, in which tool use, retry logic, planning structure, and multi-agent interaction generate emergent and potentially unstable system behavior. Fifth, it includes evidence and assurance surfaces, where the question is no longer only whether the system functions, but whether its structural condition can be justified, reconstructed, and defended under deployment, audit, or governance constraints.

SORT-AI is therefore defined by a stable Level-0 domain architecture rather than by any individual application. Applications such as AI.01, AI.04, or AI.13 instantiate recurrent structural problem forms within the domain, but they do not exhaust it. The domain is organized by a persistent relation between problem space, structural classes, recurrent application forms, and diagnostic dimensions, with an inner regime structure inside the application layer. This architecture is what makes the domain reusable and extensible across subsequent research and decision-oriented work in the AI domain.

2.2. Independence from Individual Whitepaper Versions

This manuscript is intended to be mathematically and conceptually self-contained at the AI-domain interpretation level. It does not require prior reading of a particular SORT whitepaper version in order to understand what SORT-AI is, how it organizes advanced AI systems, how it performs structural diagnosis, and on what minimum mathematical basis the domain reading rests.

At the same time, self-containment at the domain level does not imply duplication of the full core framework. The deeper derivation of the operator algebra, kernel calibration, admissibility conditions, and validation tolerances remains part of the canonical SORT whitepaper line. Those references serve as the deeper mathematical archive for the underlying Level-0 structural machinery. The present manuscript instead provides the minimum mathematical structure required to read AI systems through SORT-AI and to understand how structural diagnosis operates in the AI domain, as developed later in Section 9.

This distinction is important for long-term stability. The role of the domain paper is to remain readable and reusable even if subsequent core-framework documents are extended, reformulated, or clarified. Future framework revisions can therefore be absorbed through citation and consistency alignment without requiring that the domain paper itself be rewritten every time the deeper core documentation evolves. In this way, the domain paper serves as the canonical architectural layer for SORT-AI, while the whitepapers remain the deeper mathematical reference layer.

 doi:10.20944/preprints202512.1334.v4

2.3. The Role of SORT-AI in the Broader SORT Framework

SORT-AI is one technical domain within the broader SORT framework [20] and shares the same mathematical core as other domain modules such as SORT-CX, SORT-QS, and SORT-COSMO. The underlying framework is therefore not redefined separately for AI. What changes from domain to domain is not the formal substrate itself, but the way in which structural states are interpreted, the types of coupling that dominate the system class, and the kinds of evidence surfaces that become relevant for diagnosis and decision-making.

Domain differentiation arises through three factors. The first is interpretation: the same formal machinery is applied to different classes of systems, and the meaning of structural states depends on the domain in which they are read. The second is coupling structure: AI systems exhibit characteristic interactions between models, runtimes, control planes, evaluation surfaces, and deployment environments that differ from the characteristic couplings of complex systems or quantum systems. The third is evidence surface: the kinds of decisions supported by the domain, and the kinds of justification required, differ across domains even when the underlying structural logic remains shared.

Within this broader structure, AI is treated here as a technical domain with its own cluster architecture and application space. It is therefore neither a mere example of the general framework nor an isolated special case. Rather, it is a domain-specific realization of the shared SORT core, organized in a way that preserves cross-domain consistency while remaining sufficiently specific to support infrastructure analysis, runtime diagnosis, agentic system interpretation, and evidence-oriented deployment reasoning. The shared mathematical core developed later in Section 9 is what guarantees that this domain-specific interpretation remains compatible with the broader SORT framework rather than fragmenting into a separate formalism.

3. The Four Structural Axes of SORT-AI

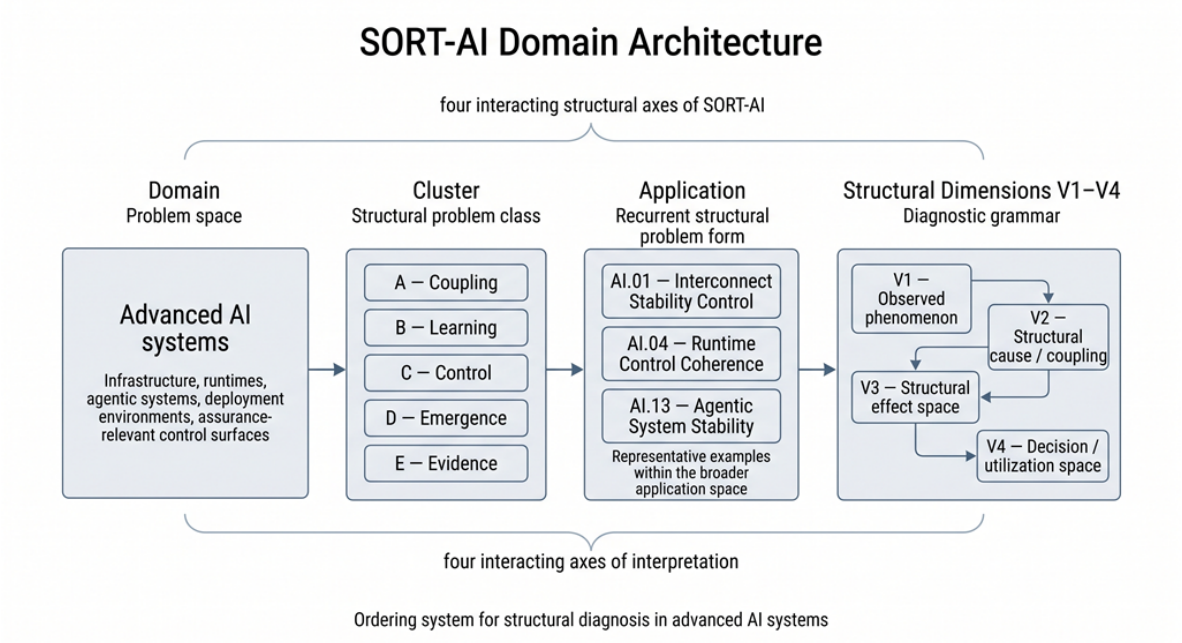

Figure 1. SORT-AI domain architecture. The framework organizes structural analysis across four interacting main axes: Domain, Cluster, Application, and Structural Dimensions V1 to V4. Domain specifies the problem space, Cluster specifies the structural problem class, Application identifies recurrent structural problem forms, and V1 to V4 define the diagnostic grammar through which observed phenomena, structural causes, effect spaces, and decision spaces are interpreted. Below the application level, Scenario Classes, Metric Sets, and a Regime Classification (core, boundary, overlap) refine the inner reading logic of the application without constituting additional main axes.

 doi:10.20944/preprints202512.1334.v4

The canonical architecture of SORT-AI is organized along four main structural axes: Domain, Cluster, Application, and Structural Dimensions V1 to V4. Together they define the ordering logic through which advanced AI systems become diagnostically readable as structurally coupled systems. Domain identifies the system class, Cluster the dominant structural regime, Application the recurrent problem form, and V1 to V4 the diagnostic sequence from observation to decision surface.

This four-axis structure is necessary because no single ordering principle is sufficient on its own. Domain alone would remain too coarse, Application alone would produce a growing list without stable architecture, and diagnostic dimensions alone would not fix either system class or recurrent problem form. The four axes therefore operate together as the structural backbone of the AI domain.

Below the application level, the architecture admits a further diagnostic decomposition:

$$\text{Application} \longrightarrow \text{Scenario Class} \longrightarrow \text{Metric Set} \longrightarrow \text{Regime Classification}.$$

This decomposition is not a fifth main axis. Scenario Class, Metric Set, and Regime Classification are inner reading layers of the application: they refine how a single application is read structurally without competing with Domain, Cluster, Application, or V1 to V4. The four main axes therefore remain the architectural backbone of SORT-AI, and the inner decomposition is developed in Section 6.4.

3.1. Domain — the What

Domain specifies the problem space to which the framework is applied. In the present manuscript, that space is the class of advanced AI systems, including infrastructure, runtimes, agentic systems, deployment environments, and assurance-relevant control surfaces. The domain axis therefore determines what kind of system is being analyzed and within which operational and evidentiary context the analysis is meaningful.

Its methodological role is to fix the interpretive frame through which structural states become meaningful. The mathematical basis of SORT is shared across domains, but the reading of structural states depends on the problem class to which it is applied.

3.2. Cluster — the How

Cluster specifies the structural problem class in which a phenomenon is situated. Within SORT-AI, the current cluster architecture consists of five classes: Coupling, Learning, Control, Emergence, and Evidence. These are not merely organizational categories. They separate structurally distinct regimes of system behavior.

The cluster axis therefore identifies how the relevant problem takes structural form. Coupling addresses interdependence, Learning temporal adaptation, Control operative coordination, Emergence non-linear or self-amplifying behavior, and Evidence traceability and justification. In this way, Cluster functions as the bridge between domain scope and concrete application forms.

3.3. Application — the Example

Application specifies a recurrent structural problem form within a cluster. It is not an arbitrary use case, but a named and reusable structural pattern through which the architecture becomes operational. Examples include AI.01, AI.04, and AI.13, each of which instantiates a distinct recurrent condition within a broader structural regime.

The application axis also makes the domain extensible. New applications can be added when recurrent structural conditions emerge that are not yet captured by the existing catalog. In this way, the domain remains stable even as the application layer grows.

3.4. Structural Dimensions V1 to V4 — the Why

Structural Dimensions V1 to V4 define the diagnostic grammar of SORT-AI. While Domain, Cluster, and Application identify the system class, structural regime, and recurrent problem form, V1 to V4 determine how diagnosis proceeds from observation to decision relevance.

 doi:10.20944/preprints202512.1334.v4

V1 specifies the observed structural phenomenon, V2 the structural cause or coupling, V3 the structural effect space in which the condition becomes intelligible, and V4 the decision and utilization space that follows from that reading. Their importance lies in repeatability: the same grammar can be applied across clusters and applications without changing its internal logic. V1 to V4 therefore function as the invariant diagnostic sequence through which the domain is made scientifically readable.

4. SORT-Sovereign as a Meta-Domain Layer

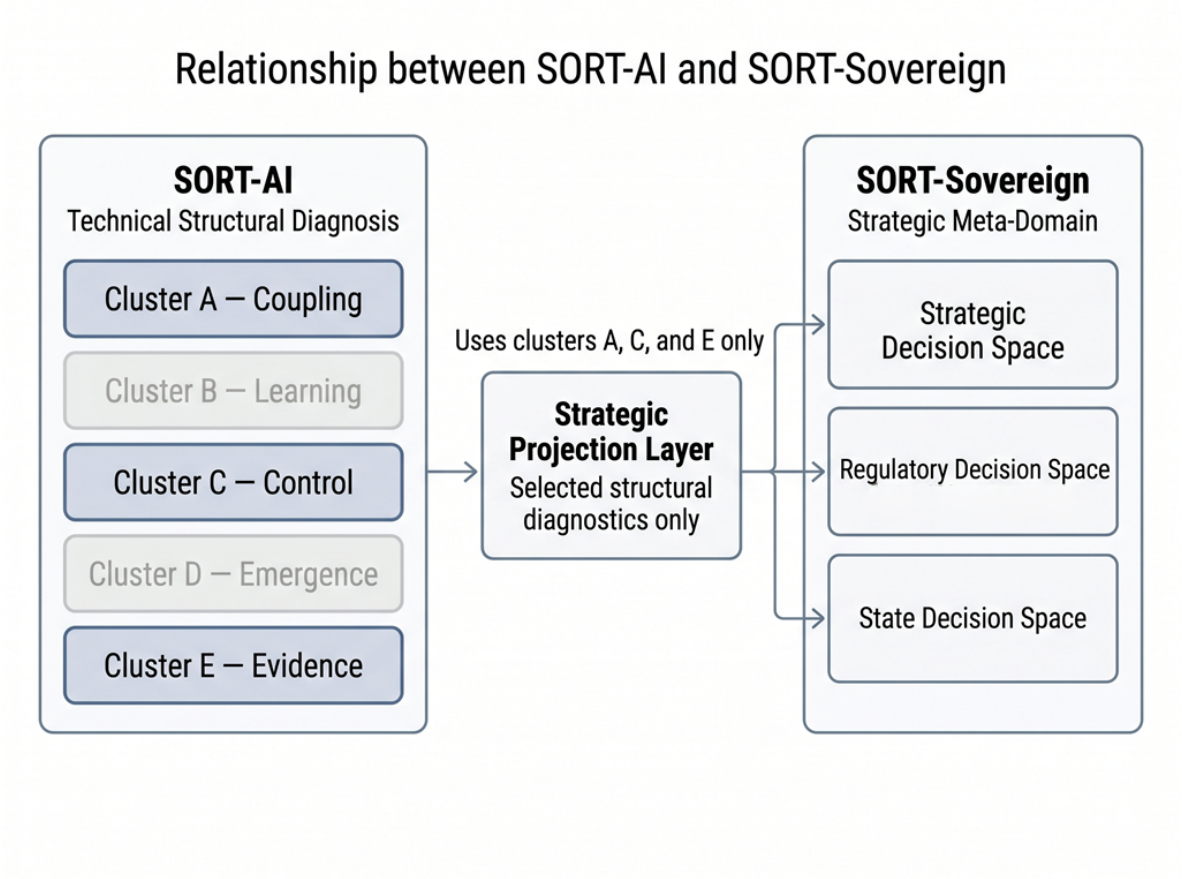

Figure 2. Relationship between SORT-AI and SORT-Sovereign. SORT-AI provides the technical structural diagnosis, whereas SORT-Sovereign projects that diagnosis into strategic, regulatory, and state decision spaces. Sovereign uses only clusters A, C, and E.

SORT-Sovereign is introduced in this paper as the meta-domain layer that connects technical structural diagnosis to decision spaces beyond engineering. The need for such a layer follows directly from the practical role of advanced AI systems. Once frontier or large-scale AI systems are deployed in regulated, public, or strategically consequential environments, the relevant questions are no longer exhausted by whether a system is functionally performant. They extend to whether the system remains auditable, governable, procurable, defensible, and controllable under institutional conditions that require more than technical adequacy. In this sense, the role of SORT-Sovereign is not to replace technical diagnosis, but to translate technical structural findings into decision surfaces that become meaningful for regulatory, procurement, strategic, and state actors. SORT-Sovereign is therefore a decision-relevance projection of technical structural findings; it is not a compliance-certification, assurance-certification, or conformity-assessment mechanism, and it does not issue regulatory determinations.

This distinction is especially important for the present domain paper. SORT-AI establishes the technical domain architecture through which coupled AI systems are read structurally. SORT-Sovereign establishes the meta-domain architecture through which those structural findings become legible

 doi:10.20944/preprints202512.1334.v4

beyond the engineering frame. Without such a meta-layer, AI-domain results would remain bounded by technical interpretation even when their practical consequences clearly extend into governance, assurance, and strategic infrastructure decision-making. With it, the framework gains a second level of interpretability in which technical findings can be projected into external decision spaces without requiring a different technical formalism.

The inclusion of SORT-Sovereign therefore clarifies a central architectural point of the broader SORT framework. Technical domains such as AI, CX, and QS diagnose structural conditions within their respective system classes. The Sovereign layer does not add a new technical substrate to those domains. Rather, it reads selected outputs of those domains in terms of strategic consequence, regulatory visibility, and state-relevant decision logic. This paper introduces that relation in compressed form because the AI domain cannot be fully understood as a scientific architecture if its evidence and control surfaces are interpreted only internally.

4.1. Why SORT-Sovereign Exists

Technical structural findings from AI, CX, and QS increasingly have direct consequences beyond engineering, including strategic planning, regulatory evaluation, procurement decisions, and state-relevant assessments [21,22]. This is particularly evident in advanced AI systems, where runtime behavior, evidence gaps, evaluation opacity, and deployment drift may remain manageable inside a technical organization while simultaneously becoming problematic in institutional settings that require formal auditability, defensible control logic, and strategic accountability.

The need for a higher-order reading of such conditions is also reflected in the wider AI safety and oversight literature. Several lines of work have emphasized that the central difficulty in advanced AI governance lies not merely in the properties of isolated models, but in whether system behavior remains observable, interpretable, and controllable under realistic deployment and oversight conditions [23–27]. These are not purely model-internal questions. They are system-level questions that depend on the relation between technical behavior and institutional decision structures.

SORT-Sovereign exists to read this relation systematically. It provides a meta-domain layer that projects technical structural results into strategic, regulatory, and state decision spaces without introducing a separate technical formalism. In doing so, it preserves continuity with the technical domains while making their outputs usable in contexts where the relevant question is not only whether a system works, but whether its structural condition can be governed, justified, and acted upon.

4.2. Relation Between SORT-AI and SORT-Sovereign

SORT-AI and SORT-Sovereign occupy different levels of the same architectural stack. SORT-AI is a technical domain. It diagnoses structural conditions in advanced AI systems, including coupling effects, runtime incoherence, learning-related drift, emergent agentic dynamics, and evidence surfaces. Its analytical object is the AI system itself as a coupled technical structure. SORT-Sovereign, by contrast, is a meta-domain. It does not diagnose a distinct technical substrate. Instead, it projects technical findings from AI, CX, and QS into decision spaces in which strategic, regulatory, or state consequences become primary.

The distinction is therefore one of level rather than of competing scope. SORT-AI answers questions such as whether a runtime regime is structurally coherent, whether a deployment surface exhibits drift, or whether an agentic workflow is entering an unstable regime. SORT-Sovereign answers questions such as whether those same technical conditions are auditable, strategically governable, or defensible under external requirements. The former is concerned with technical structure. The latter is concerned with the decision relevance of that structure once it is situated in governance-bearing environments.

A Sovereign reading is thus defined only in conjunction with a technical domain reading. It does not stand alone, because it does not generate its own technical evidence independently of the underlying domain. Instead, it reinterprets domain-specific structural findings at the level of institutional action. This dependence is not a weakness of the meta-domain architecture. It is the

 doi:10.20944/preprints202512.1334.v4

reason it is analytically useful: it preserves technical fidelity while extending structural diagnosis into non-technical decision spaces.

4.3. Sovereign Cluster Scope — A, C, E

SORT-Sovereign uses only clusters A, C, and E. This restricted cluster scope follows from the kind of decision spaces the meta-domain addresses. Cluster A, Coupling, is relevant because strategic and state-level questions often turn on infrastructure dependence, bottleneck concentration, systemic fragility, and cross-layer interdependence. Cluster C, Control, is relevant because institutional decision-makers require visibility into whether technical systems remain controllable, whether control logic is coherent, and whether runtime behavior can be governed under stress or policy constraints. Cluster E, Evidence, is relevant because auditability, traceability, assurance, and formal defensibility are indispensable once technical systems enter procurement, compliance, or sovereign deployment settings.

By contrast, Clusters B and D are excluded from the Sovereign scope. Learning and Emergence remain central within the technical domains, but they do not project as directly into strategic and regulatory decision surfaces without first being rendered through coupling, control, or evidence considerations. This does not mean that learning drift or emergent behavior are irrelevant to Sovereign reasoning. It means that their relevance becomes institutionally legible only after they have been transformed into questions of dependency, controllability, or evidence sufficiency.

The selective use of A, C, and E therefore reinforces the role of SORT-Sovereign as a meta-domain rather than a second technical domain. It does not replicate the full internal richness of the technical layer. Instead, it identifies the subset of structural classes through which technical diagnosis becomes institutionally actionable.

4.4. Miniature Use Case — Sovereign Runtime Auditability for Frontier AI Deployment

A compact use case helps clarify why the meta-domain is necessary. Consider a frontier AI deployment that is functionally stable in the narrow technical sense. The system serves its workload, meets operational thresholds most of the time, and may even appear acceptable under standard observability metrics. Yet when the same system is examined under regulatory or sovereign deployment requirements, a different question emerges: can the runtime control behavior be reconstructed, justified, and defended if challenged by an auditor, regulator, procurement body, or public-sector review process?

The technical basis of this case is provided by Runtime Control Coherence, represented by AI.04. At the technical level, AI.04 asks whether independently correct control layers remain globally coherent when composed under scale. This produces a structural reading of the runtime that is already richer than conventional monitoring. However, this technical reading still does not answer whether the resulting system is institutionally governable. The gap between operational sufficiency and formal auditability remains open.

SORT-Sovereign closes that gap by projecting the runtime reading into a Sovereign evidence surface. In the present use case, SOV.03, Sovereign Runtime Auditability and Control Transparency, functions as the corresponding meta-domain application. It does not replace the runtime diagnosis performed by AI.04. Rather, it translates the technical finding into a form in which regulatory and state actors can ask whether control integrity is demonstrable, whether evidence exists without full implementation disclosure, and whether the deployment remains defensible in a sovereign or high-assurance context. This miniature use case is intentionally compressed. Its function is not to present a full use-case paper, but to show why the meta-domain is structurally necessary.

 doi:10.20944/preprints202512.1334.v4

4.5. V1 to V4 Reading of the Sovereign Mini-Use-Case

Table 2. V1 to V4 reading of the Sovereign Runtime Auditability mini-use-case (`SOV.03` combined with `AI.04`).

Dim.	Reading
V1	Frontier AI deployment is functionally stable yet not formally auditable; runtime control behavior cannot be reconstructed for regulators or state actors.
V2	Gap between operational control, effective scheduling and policy enforcement, and demonstrable control integrity, namely an evidence surface suitable for auditors.
V3	SORT-AI reading of the runtime through AI.04 provides an architectural evidence layer rather than an implementation-level reconstruction; SORT-Sovereign projects this layer into a regulator-compatible decision space.
V4	Decisions become possible in three areas: regulatory compliance posture, audit-readiness for assessments, and procurement or exit strategy for sovereign AI deployments.

The V1 to V4 reading of this miniature case shows that the Sovereign layer does not invent a new problem. It reformulates an already existing structural condition so that it becomes decision-relevant beyond engineering. V1 identifies the externally visible paradox: functional stability without formal auditability. V2 identifies the structural cause of that paradox: the gap between operational control and demonstrable control integrity. V3 shows how the technical reading generated by AI.04 is transformed into an evidence-bearing surface suitable for institutional interpretation. V4 identifies the decision areas that become newly accessible once this projection is made.

This is the central role of the meta-domain. It does not add a second diagnosis to the same technical system. It changes the level at which the diagnosis becomes meaningful. The same structural condition can therefore be read technically through SORT-AI and meta-structurally through SORT-Sovereign without generating a contradiction between the two.

4.6. Why This Matters for the AI Domain

The inclusion of the Sovereign layer clarifies why the evidence surface of SORT-AI matters beyond engineering. Without a Sovereign projection layer, structural findings in AI would remain technical artifacts whose significance would end at the boundary of technical interpretation. They could describe drift, coherence loss, coupling instability, or evidence insufficiency, but they would not yet explain why those conditions matter in procurement, regulation, public accountability, or sovereign infrastructure strategy.

With the Sovereign layer in place, the AI domain acquires a second form of relevance. Its findings become inputs to governance-relevant decision surfaces rather than remaining confined to internal technical optimization. This is particularly important for advanced AI systems whose deployment contexts extend into regulated sectors, critical infrastructure, public administration, or state-sensitive digital environments. In such settings, technical adequacy alone is not sufficient. The system must also be governable, auditable, and strategically interpretable.

For the architecture of this paper, the implication is direct. SORT-AI can only be understood fully if its evidence and control surfaces are read not only as technical conditions, but also as potential inputs to meta-domain projection. The Sovereign layer therefore belongs near the beginning of the manuscript, before the detailed cluster architecture of the AI domain, because it clarifies from the outset that the significance of structural diagnosis does not end with technical interpretation.

5. Cluster Architecture of the AI Domain

The AI domain of SORT-AI is organized through five clusters: Coupling, Learning, Control, Emergence, and Evidence. These clusters are not merely topical groupings. They define the primary structural regimes through which advanced AI systems become diagnostically readable and form the middle layer between the abstract domain definition and the concrete application space.

 doi:10.20944/preprints202512.1334.v4

This architecture is necessary because advanced AI systems do not fail or degrade in one uniform way. Instability may arise from cross-component coupling, temporal adaptation, incoherent control logic, emergent agentic amplification, or the inability to generate defensible evidence about the system at all. The cluster structure separates these dominant problem classes while preserving their relation within a common architecture.

The cluster system also stabilizes the domain as the application space grows. New applications do not enter arbitrarily. They must arise within one of the existing structural regimes, or motivate a future architectural extension if a genuinely new class appears.

5.1. Overview of the Five AI Clusters

The five AI clusters correspond to five recurrent structural regimes in advanced AI systems under real deployment conditions. Cluster A, Coupling, captures conditions in which physical, logical, or topological interdependence becomes the primary driver of system behavior. Cluster B, Learning, captures conditions shaped by temporal adaptation, training dynamics, and distributional shift. Cluster C, Control, captures conditions in which the dominant problem lies in operative coordination and runtime coherence across multiple decision layers. Cluster D, Emergence, captures conditions in which non-linear, self-amplifying, or agentic behavior becomes structurally salient. Cluster E, Evidence, captures conditions in which traceability, auditability, and structural justification become central to interpretability and governability.

This five-part structure is an analytical separation, not a claim that real systems instantiate only one cluster at a time. Cross-cluster phenomena are common, but the role of the cluster system is to define the dominant structural regime through which a condition first becomes readable. The current public SORT-AI application space comprises 52 applications distributed across these five clusters, with the strongest concentration currently in Coupling, Control, and Emergence.

The choice of five clusters is not arbitrary. Each cluster corresponds to a dominant structural regime already treated as a distinct concern in the surrounding systems and AI literature: physical and topological coupling in large-scale infrastructure [1,2], temporal adaptation and distributional shift in learning systems [28,29], operative control and runtime coordination in serving and orchestration [5,30], emergent and agentic amplification in multi-step execution [31,32], and traceability or assurance in governance-bearing settings [21,22]. The clusters are entry regimes rather than mutually exclusive ontological categories: they identify where a structural condition first becomes legible, while the cross-cluster transitions of Section 14 describe how a condition can propagate between them.

5.2. Cluster A — Coupling

Cluster A addresses structural conditions in which physical, logical, or topological coupling between components or layers becomes the dominant determinant of system behavior. The relevant degradation or instability cannot be reconstructed by inspecting components in isolation. It arises from the way components are joined through communication paths, memory dependencies, synchronization constraints, routing surfaces, or other forms of structural interdependence.

Within advanced AI systems, this cluster includes interconnect instability, memory-path coupling, distributed serving asymmetries, service-mesh interaction, and other forms of infrastructure-level or topology-sensitive performance collapse [1,2,33]. Even when throughput, bandwidth, or compute capacity appear sufficient in local terms, the composed behavior of the system may still degrade because structural coupling redistributes pressure in ways that local metrics cannot resolve.

The guiding question of Cluster A is therefore: how do component interactions produce system-level behavior that local metrics cannot resolve?

5.3. Cluster B — Learning

Cluster B addresses structural conditions in which temporal adaptation becomes the dominant driver of system behavior. The decisive question is not only how a system behaves under load

 doi:10.20944/preprints202512.1334.v4

or composition, but how its structural condition changes through training, fine-tuning, continual adaptation, benchmark exposure, or deployment-side learning dynamics.

In advanced AI systems, this cluster includes benchmark integrity, evaluation-context instability, training-deployment divergence, and learning-related drift under changing data or environment conditions [3,28,29,34]. These conditions matter because AI systems are often judged through static evaluation surfaces even though their operational significance depends on how learning processes modify the relation between model behavior, test conditions, and deployment realities.

The guiding question of Cluster B is therefore: how does learning over time interact with static evaluation surfaces?

5.4. Cluster C — Control

Cluster C addresses structural conditions in which the dominant problem lies in operative coordination, runtime coherence, and the interaction of multiple decision structures. Its defining assumption is that local correctness of control mechanisms does not imply global coherence of the system they jointly govern.

This cluster is especially central for modern AI systems because operational performance increasingly depends on how multiple control surfaces interact rather than on the existence of any single controller in isolation. Examples include runtime control coherence, inference pipeline coherence, scheduler–orchestrator interaction, and related runtime-level coordination problems [8,30,35]. In such environments, the primary structural question is not whether a given controller is broken, but whether multiple controllers acting on overlapping system surfaces remain mutually coherent over time.

The guiding question of Cluster C is therefore: when do locally correct control loops produce globally incoherent behavior?

5.5. Cluster D — Emergence

Cluster D addresses structural conditions in which non-linear, self-amplifying, or agentic behavior becomes the dominant analytical problem. The core assumption of this cluster is that the composition of correct components may still generate system behavior whose practical significance exceeds the explanatory adequacy of local-component reasoning.

Within advanced AI systems, this includes agentic system instability, multi-agent regime interaction, retry amplification, tool-mediated escalation, and other conditions in which composed behavior becomes self-reinforcing or qualitatively shifts under scale or interaction [27,31,32]. These phenomena are especially important in systems involving autonomy, iterative planning, chained tool use, or recursive verification logic.

The guiding question of Cluster D is therefore: under what conditions does composition of correct components produce incoherent or self-amplifying behavior?

5.6. Cluster E — Evidence

Cluster E addresses structural conditions in which traceability, auditability, and structural justification become primary. Unlike the preceding clusters, which focus on how systems behave, Cluster E focuses on whether the structural condition of a system can be rendered evidentially legible without requiring exhaustive disclosure of internal implementation details.

In advanced AI systems, this cluster includes structural stability evidence packs, deployment drift signal aggregation, and the auditability of complex control planes and deployment surfaces [21,22,36–38]. These conditions become increasingly important in regulated sectors, high-assurance settings, sovereign deployments, and any context in which operational claims must be translated into formal evidence.

The guiding question of Cluster E is: what structural evidence can be generated without disclosing implementation details?

 doi:10.20944/preprints202512.1334.v4

5.7. Current AI-Domain Distribution of Applications

Table 3. Current distribution of SORT-AI applications across clusters. The AI domain currently comprises 52 applications.

Cluster	Structural focus	Application count
A — Coupling	Physical and logical coupling	26
B — Learning	Temporal adaptation and learning	6
C — Control	Operative control and coherence	9
D — Emergence	Emergent, non-linear behavior	10
E — Evidence	Traceability and auditability	1
Total		**52**

The current distribution reflects the present state of the public SORT-AI application catalog. The strongest concentration lies in Coupling, Control, and Emergence, which is consistent with the main operational pressure points of current AI infrastructures and agentic systems. By contrast, Learning and Evidence currently occupy a smaller public footprint, but this should not be read as lower importance. In particular, the small public count of Cluster E does not imply low strategic relevance, as the discussion of the Sovereign meta-domain in Section 4 makes clear.

The current distribution should therefore be understood as contingent rather than fixed. The architecture of the domain remains stable, but the application counts within clusters may evolve through additive extension over time.

6. Applications as Derived Structural Problem Forms

Applications are the layer at which the domain architecture becomes operational. They do not replace the domain or cluster structure; they specify the point at which recurrent structural conditions become concrete enough to be named, compared, reused, and extended. In this sense, an application is neither an arbitrary label nor a deployment-specific description. Applications are recurrent structural problem forms and structured regime spaces: each application identifies a recurrent structural condition within a given cluster and, at the same time, admits an inner decomposition into Scenario Classes, Metric Sets, and a Regime Classification through which that condition becomes diagnostically readable in finer detail. The hierarchy used throughout this manuscript is therefore:

$$\text{Domain} \rightarrow \text{Cluster} \rightarrow \text{Application} \rightarrow \text{Scenario Class} \rightarrow \text{Metric Set} \rightarrow \text{Regime Classification}.$$

This role is important because the domain would otherwise remain too abstract for repeated use. Domain identifies the problem space, and clusters identify the dominant structural regimes, but it is the application layer that makes the framework reusable across concrete analytic settings. Applications provide the unit through which subsequent technical and decision-oriented analyses can be anchored without losing the architectural consistency of the overall domain.

At the same time, the application layer must remain disciplined. If applications were treated as free-form examples, the domain would gradually fragment into a loose collection of case descriptions. The purpose of this section is therefore to define what an application is, how it emerges, how its identity relates to the diagnostic dimensions V1 to V4, and how its inner regime structure is read. The result is a stable application layer that is concrete enough for operational use and structured enough for long-term extension.

6.1. Applications Are Not Isolated Use Cases

Applications should not be understood as isolated use cases or ad hoc categorizations. A use case typically describes a context-specific scenario in which a system is deployed, observed, or evaluated. An application, by contrast, is a recurrent structural problem form. It identifies a condition that can

 doi:10.20944/preprints202512.1334.v4

appear across multiple contexts while preserving the same underlying diagnostic logic. This distinction is essential for the architecture of SORT-AI, because the framework is meant to organize recurring structural classes rather than accumulate unrelated examples.

Each application therefore instantiates a structural condition within a specific cluster. AI.04, for example, is not merely a runtime case; it is the recurrent control-class problem form of Runtime Control Coherence. AI.01 is not merely an infrastructure example; it is the recurrent coupling-class problem form of Interconnect Stability Control. AI.13 is not merely an agentic scenario; it is the recurrent emergence-class problem form of Agentic System Stability. What makes these applications reusable is precisely that they capture recurring structural conditions rather than one-off observations.

This distinction also explains why applications can support multiple scenarios without collapsing into them. A single application may be instantiated in several different operational environments, but those scenarios remain realizations of one structural problem form rather than separate applications in their own right. The application layer therefore stabilizes the relation between structural recurrence and contextual variation. It is the place where the framework becomes concrete without becoming fragmented.

6.2. How New Applications Emerge

New applications emerge when a structural phenomenon recurs across contexts but is not yet adequately covered by an existing application. The criterion is therefore not novelty in the everyday sense, but structural recurrence combined with diagnostic insufficiency of the current application set. If a phenomenon can already be read coherently within an existing application, the framework does not require a new one. If, however, a recurring structural condition appears whose dominant regime, coupling logic, or decision surface is not adequately captured, then a new application becomes justified.

This criterion can be stated as an explicit application-admission rule. A new application is admitted only if it satisfies all of the following: (i) recurrent structure, meaning that the condition appears across multiple contexts rather than as a single instance; (ii) a stable metric signature, meaning that it is associated with a declared metric set that reads consistently across those contexts; (iii) independent diagnostic relevance, meaning that it is not reducible to an existing application without loss of diagnostic content; and (iv) V1 to V4 readability, meaning that it admits a coherent reading through the diagnostic grammar. A single overlap scenario between two existing applications does not by itself define a new application. Without these four properties, the condition is documented at the scenario-class or overlap-regime level while existing application identities are preserved.

This process is governed by the additive-extension rule. The existing application structure is not revised by deleting, replacing, or redefining previously established applications. Instead, the application space grows through supplementation. This preserves long-term architectural stability while allowing the domain to adapt to new structural conditions as AI systems evolve. The additive-extension rule is therefore not merely a catalog management principle. It is an architectural principle that protects the continuity of the domain over time.

The importance of this rule is scientific as well as organizational. If existing applications could be rewritten whenever new phenomena appeared, the domain would lose its stability as a canonical reference architecture. Later papers would no longer be able to rely on a persistent application layer. By contrast, additive extension allows the framework to grow while preserving the interpretive consistency of earlier work. This makes the application architecture suitable for a research program that is expected to expand through new papers, new scenarios, and new classes of AI systems without forfeiting continuity.

6.3. Application Identity and Diagnostic Dimensions

Every application is identified by three primary elements: an application ID, a title, and a cluster assignment. The ID provides the stable referential anchor, the title provides the diagnostic name of the structural problem form, and the cluster assignment places that problem form within the dominant

 doi:10.20944/preprints202512.1334.v4

structural regime of the domain. Together, these elements define the primary identity of the application. They specify what structural condition is being named and where it belongs in the domain architecture.

The diagnostic dimensions V1 to V4 do not replace that identity. They are supplementary diagnostic fields that explain how the application is read. V1 identifies the observed structural phenomenon. V2 identifies the structural cause or coupling. V3 identifies the structural effect space through which SORT renders the condition intelligible. V4 identifies the decision and utilization space opened by that reading. In this way, the application identity specifies what the recurring problem form is, while V1 to V4 specify how that problem form becomes diagnostically meaningful.

This distinction is necessary to keep the architecture ordered. If V1 to V4 were treated as primary identifiers, then the application layer would dissolve into a set of descriptive fields without stable structural names. Conversely, if applications were treated only as names without diagnostic dimensions, the framework would lose its repeatable interpretive logic. Application identity and diagnostic dimensions therefore function together but at different levels. The application states the recurrent structural form. V1 to V4 state the diagnostic sequence through which that form is read.

6.4. Scenario-Class Architecture within Applications

Applications are not only recurrent structural problem forms; they are also internally structured regime spaces. Within a single application, the same recurrent problem form may appear under distinct structural conditions that differ enough to require their own diagnostic vocabulary while still belonging to one application identity. The Scenario-Class Architecture introduced in this manuscript makes this inner structure explicit and connects it to structural assessment.

A Scenario Class is a typed manifestation of an application under a specific structural condition. It is not a new application and not a deployment-specific use case. A Scenario Class is a controlled structural slice of the application that admits its own targeted reading while preserving application identity. A Metric Set is a declared group of observable or derived indicators through which a Scenario Class is read structurally; it is the layer at which Scenario Classes become structurally assessable. A Regime Classification situates each Scenario Class as one of three regime types. A Core Regime is a central manifestation of the application that expresses its axis-defining structural mode. A Boundary Regime is a limit case in which the application approaches capacity, control, context, SLA, structural, or validity boundaries while remaining within its own structural axis. An Overlap Regime is a mixed regime in which two applications interact within a single Scenario Class and a structural transition between them becomes visible.

Overlaps are not failures of the taxonomy; they are discovery signals. They mark the points at which the coupled behavior of advanced AI systems crosses application boundaries without leaving the domain architecture, and they often expose where structurally important phenomena would otherwise be misread as defects of one application or as undifferentiated mixtures of several. At the same time, an overlap is not by itself a license to introduce a new application. A candidate application emerging from an overlap regime requires recurrent structure across contexts, a stable metric signature, independent diagnostic relevance, and V1 to V4 readability. In the absence of these properties, the overlap is documented at the scenario-class level and the existing application identities are preserved.

The Scenario-Class Architecture therefore extends the application layer from within rather than competing with it. It connects application identity to structural assessment by making explicit which manifestations of an application are read, through which indicators, and under which regime classification. A reproducible Core-3 kernel-damping evidence protocol that exercises this inner architecture for AI.01, AI.04, and AI.13 is documented in a companion technical note [19]; the present manuscript fixes only the architectural and diagnostic role of Scenario Classes, Metric Sets, and Regime Classifications within the SORT-AI domain.

 doi:10.20944/preprints202512.1334.v4

7. Structural Dimensions V1 to V4 as Diagnostic Lens

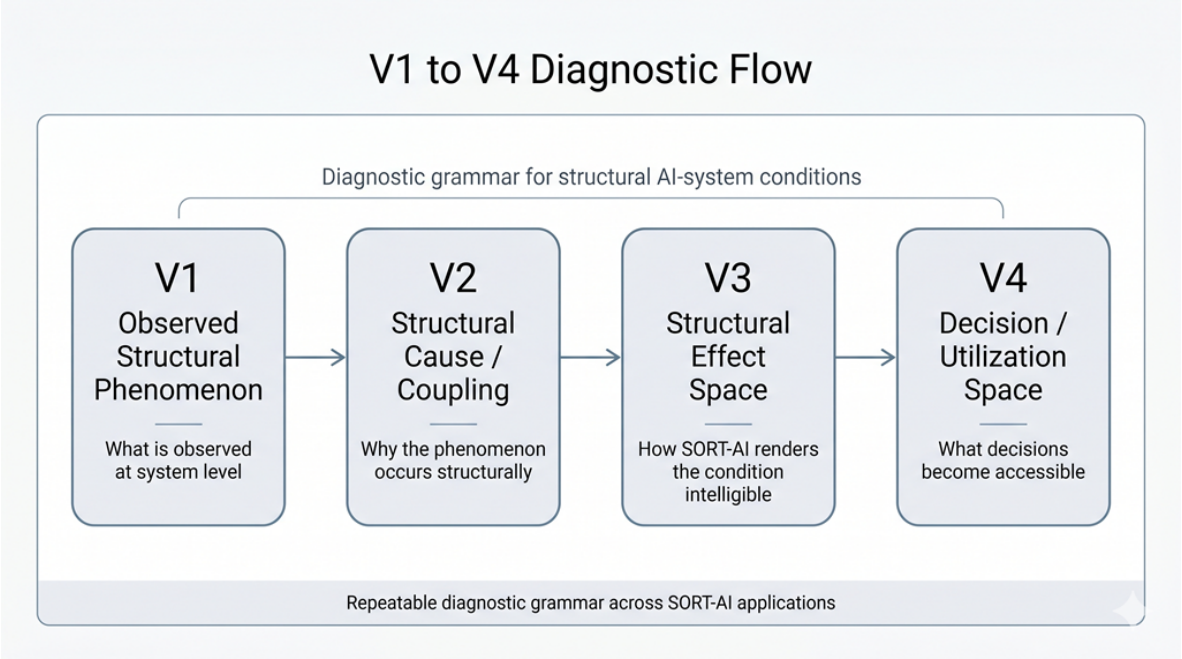

Figure 3. The V1 to V4 diagnostic flow used throughout the SORT-AI domain. V1 identifies the observed structural phenomenon. V2 identifies the structural cause or coupling. V3 identifies the structural effect space through which SORT-AI renders the condition intelligible. V4 identifies the decision and utilization space that becomes accessible through that reading.

The Structural Dimensions V1 to V4 define the diagnostic grammar of SORT-AI. If Domain specifies the problem space, Cluster specifies the structural regime, and Application specifies the recurrent structural problem form, then V1 to V4 specify how that problem form is read. They are therefore not an optional descriptive layer added after the architectural structure has already been fixed. They are the repeatable sequence through which structural diagnosis proceeds from observation to explanation and from explanation to decision relevance.

This diagnostic grammar is necessary because the domain architecture alone does not yet tell the reader how a structural condition becomes analytically meaningful. An application name identifies a recurrent problem form, but it does not by itself specify the path from what is observed to what becomes decision-relevant. V1 to V4 provide that path. In doing so, they make the framework operationally reusable across very different technical conditions, from interconnect instability to runtime incoherence, benchmark drift, emergent agentic amplification, and evidence-oriented audit surfaces.

The importance of V1 to V4 lies in their invariance across the domain. Their content changes from case to case, but their logical role does not. The same four dimensions can be used to read AI.01, AI.04, and AI.13, even though those applications belong to different clusters and diagnose different structural conditions. This repeatability is what allows SORT-AI to function as a scientific domain architecture rather than as a series of unrelated case descriptions.

7.1. V1 — Observed Structural Phenomenon

V1 identifies the observed structural phenomenon. It names what appears at the system level before the structural mechanism responsible for it has been established. In this sense, V1 is the diagnostic entry point. It captures the outwardly visible or operationally salient pattern that calls for explanation. Typical examples include cost escalation under nominally stable metrics, latency asymmetry under healthy local telemetry, degraded effective throughput despite sufficient nominal capacity, or growing deployment instability under apparently successful completion rates.

 doi:10.20944/preprints202512.1334.v4

The analytical role of V1 is deliberately limited. It is not yet a diagnosis, and it is not the main explanatory target of the framework. A V1 description states what is happening, but not why it is happening. Its value lies in making the structural condition explicit enough to initiate a diagnostic reading. This is why V1 remains close to observed system behavior, even when the ultimately relevant structural mechanism lies at a different level of abstraction.

The distinction between V1 and the later dimensions is methodologically important. Many operational discussions of AI systems stop at V1, especially when system performance is summarized through dashboards, benchmark outputs, success rates, or incident reports. SORT-AI begins there but does not remain there. The framework treats V1 as the point of entry into structural diagnosis, not as the level at which explanation is complete.

7.2. V2 — Structural Cause or Coupling

V2 identifies the structural cause or coupling responsible for the phenomenon observed in V1. This is the dimension at which the framework moves from appearance to composition. The question is no longer only what is visible at the system level, but which coupling relation, control interaction, transition pattern, or compositional mechanism generates the visible condition. In many cases, the central insight of SORT-AI begins here, because what appears anomalous at V1 often becomes intelligible once the relevant structural coupling has been identified.

Typical V2 patterns include circular interference between control layers, cross-layer topology coupling, retry amplification across nested system boundaries, evaluation-deployment mismatch, and agentic feedback amplification. These are not component-level defects in the ordinary sense. They are structural causes in the sense that they arise from how system elements are composed rather than from the local malfunction of any one part. V2 is therefore the first dimension at which the framework departs decisively from purely layer-local or component-local readings.

The role of V2 also explains why structural diagnosis is not reducible to monitoring. Monitoring can reveal a V1 pattern and may even localize candidate surfaces of concern, but V2 requires the articulation of the coupling logic that makes the observed condition structurally coherent as a diagnosis. This is the point at which the framework begins to identify the actual problem form rather than only describe its symptoms.

7.3. V3 — Structural Effect Space of SORT

V3 identifies the structural effect space through which SORT-AI renders the condition intelligible. This is the core analytical dimension of the framework. Once a phenomenon has been observed and a structural coupling has been identified, V3 specifies the space in which that coupling becomes readable as a structural condition. In practical terms, this includes projection states, coherence indicators, stability margins, structural drift fields, and other forms of diagnostic interpretation that are not directly available from local measurements alone.

What distinguishes V3 from V1 and V2 is that it is not merely descriptive. V1 states what is observed. V2 states what structural relation is responsible. V3 states how the condition is rendered intelligible in SORT-AI as a structural effect. This is where the framework differs most clearly from layer-local diagnostics. The problem is no longer described only as an event, anomaly, or interaction, but as a position in a structured effect space that can be compared, interpreted, and related to other problem forms within the same architecture.

For this reason, V3 is the analytical center of the diagnostic grammar. It is the dimension in which the formal and architectural elements of the framework begin to act together. Later mathematical sections will show how this dimension connects to projection, operator composition, and structural deviation. At the domain level, however, the essential point is already clear: V3 is the level at which SORT-AI turns a coupled phenomenon into a structurally legible condition.

 doi:10.20944/preprints202512.1334.v4

7.4. V4 — Decision and Utilization Space

V4 identifies the decision and utilization space that becomes accessible once the structural condition has been rendered intelligible through V3. This is the final dimension of the diagnostic grammar. Its purpose is not to prescribe a single intervention, but to identify what kinds of decisions become possible once the structural condition is understood correctly. In other words, V4 specifies the practical consequence of structural readability.

Examples of V4 outcomes include coordination-boundary redesign, architecture-level reallocation of control authority, audit-readiness decisions, procurement choices, evidence generation strategies, or exit and containment options. The specific content depends on the application and cluster in question. What remains invariant is the role of V4 as the dimension in which structural diagnosis becomes decision-relevant. Without this dimension, the framework would remain analytically interesting but practically incomplete.

V4 also clarifies why the diagnostic sequence cannot stop at V3. Structural readability alone is not sufficient if it cannot be connected to action, evaluation, or governance. The framework is not merely classificatory. It is intended to support a transition from system observation to informed structural decision. V4 is therefore not an external add-on to the framework. It is the completion of the diagnostic sequence.

7.5. V1 to V4 as a Repeatable Diagnostic Grammar

Taken together, V1 to V4 form a repeatable diagnostic grammar. Their importance lies in the fact that they apply identically across clusters, domains, and applications even though the substantive content of each dimension changes from case to case. The same grammar can therefore be used to read AI.01, AI.04, and AI.13, with each application supplying different V1 patterns, V2 couplings, V3 effect spaces, and V4 decision surfaces while preserving the same logical structure.

This repeatability is one of the reasons SORT-AI can function as a canonical domain architecture. Without a stable diagnostic grammar, each application would have to invent its own explanatory sequence, making comparison across the domain difficult and weakening the architectural integrity of the framework. V1 to V4 solve this problem by providing a common interpretive order that can be reused without being reduced to a generic checklist.

The grammar is therefore best understood as invariant in form and domain-specific in content. It remains structurally identical across applications while being filled differently depending on whether the problem concerns coupling, learning, control, emergence, or evidence. This gives the domain both rigor and flexibility: rigor, because every application is read through the same sequence; flexibility, because the content of that sequence is not artificially standardized across structurally different conditions.

V1 to V4 constitute a diagnostic ordering convention rather than an assertion of causal mechanism. The progression from observed phenomenon (V1) to structural cause or coupling (V2), effect space (V3), and decision surface (V4) orders the reading of a condition; it does not by itself establish that V2 causally produces V1 in a particular system, nor does it claim a unique causal chain. Causal interpretation, where appropriate, remains the task of application-level or empirical work. The grammar fixes how a structural condition is read in a repeatable order, not what its definitive causal explanation must be.

8. Where SORT-AI Becomes Most Informative

SORT-AI becomes most informative where the explanatory power of component-local reasoning begins to fail while the system as a whole remains operationally consequential. This occurs most clearly at structural boundaries: between model and runtime, between scheduler and orchestrator, between evaluation and deployment, between infrastructure and policy enforcement, and between technical behavior and evidence requirements. At such boundaries, locally valid observations often remain insufficient to explain the behavior of the coupled system. The resulting analytical gap is precisely the space in which structural diagnosis becomes necessary.

 doi:10.20944/preprints202512.1334.v4

The importance of this point should not be understated. Many of the most consequential AI-system problems do not first appear as explicit component failures. They appear as mismatches between local correctness and global behavior. A system may continue to function while becoming economically inefficient, strategically fragile, or operationally incoherent. In such cases, the relevant problem is not that standard diagnostic tools are useless. It is that their natural unit of observation is too local to resolve the structural condition that matters most. SORT-AI becomes informative precisely when the system must be read as a composed structure rather than as a sum of component states.

This section therefore clarifies the positive scope of the framework. SORT-AI is not meant to be maximally useful in every analytical setting. It is most useful where boundaries, composition, and multi-layer interaction dominate the behavior of the system. These are the conditions under which the diagnostic grammar introduced in the previous section becomes especially valuable, because the system exhibits patterns that are visible in operation but underexplained in conventional analytic terms.

8.1. Boundary Failures and Composition Failures

SORT-AI is most informative at structural boundaries. These include boundaries between technical layers, between control regimes, between phases of execution, and between deployment conditions that are often treated separately in practice. Such boundaries matter because they are the places at which system behavior is most likely to be shaped by translation, mediation, interference, or recomposition rather than by the local condition of any one component. The framework is therefore particularly well suited to situations in which the relevant failure is not internal to a node or module, but arises across the boundary through which multiple subsystems interact.

Boundary failures and composition failures should not be confused with ordinary component bugs. They are not defined primarily by defectiveness in isolation. They arise because coupling structure reorganizes system behavior in ways that local correctness cannot prevent [2,39,40]. A scheduler can be correct, a runtime can be correct, and an orchestration policy can be correct, while the composed system formed by their interaction still becomes unstable or inefficient. What fails in such cases is not the isolated element, but the composition relation through which multiple correct elements are joined.

This is precisely the regime in which the language of boundary failure becomes analytically stronger than the language of malfunction. Boundary failures identify structurally meaningful discontinuities between otherwise legitimate subsystems. Composition failures identify the inability of those subsystems to preserve global coherence when combined. Together, these notions define one of the principal zones in which SORT-AI becomes distinctively useful: they capture failures that are real, consequential, and recurrent, yet do not map cleanly onto the ordinary concept of a broken part.

8.2. Standard Diagnostics versus Structural Diagnostics

Standard diagnostics remain indispensable for the detection of component-local failures, performance regressions, hardware faults, local bottlenecks, and operational anomalies. They provide the first layer of observability and are often the correct tool for identifying explicit failure states. SORT-AI does not seek to replace these tools, nor does it claim that system-level structural diagnosis renders them obsolete. On the contrary, a coherent technical practice requires both kinds of analysis: local diagnostics for component integrity and structural diagnostics for coupled-system behavior.

The distinction lies in the object being read. Standard diagnostics typically ask whether an individual component, metric surface, or subsystem is functioning within expected parameters. Structural diagnostics ask whether the composed system remains coherent under the interaction of multiple such components and surfaces. This difference in analytical object is what makes the two approaches complementary rather than competitive. When the system is dominated by local defects, standard diagnostics may be sufficient. When the system is dominated by cross-layer, cross-regime, or cross-boundary effects, structural diagnostics become necessary.

 doi:10.20944/preprints202512.1334.v4

SORT-AI therefore operates as a distinct analytical layer rather than as a substitute for monitoring. It complements tracing, telemetry, benchmarking, and observability by addressing the class of conditions that those tools often expose only indirectly. The framework becomes most informative when the available measurements are internally plausible, locally consistent, and yet insufficient to explain the behavior of the system as a structural whole. That is the precise point at which the coupled system becomes the relevant unit of analysis.

8.3. Examples of AI Structural Paradoxes

One of the clearest indicators that a structural reading is needed is the appearance of an AI-system paradox: a condition in which the system behaves in a way that appears contradictory when interpreted through local metrics alone, yet becomes coherent once the relevant structural composition is identified. Such paradoxes are not rhetorical devices. They are practical signals that the explanatory frame is too narrow for the behavior being observed.

A first example is the coexistence of good local metrics with escalating system cost [14,15]. In such cases, utilization, completion rates, or local throughput may appear healthy, yet the cost of obtaining effective output continues to rise. The paradox exists because the system is judged locally while the economic burden is generated structurally. A second example is the coexistence of locally correct control loops with incoherent global behavior [30,40]. Here the paradox lies in the fact that no single control layer is malfunctioning, yet the combined system exhibits instability, oscillation, or waste.

A third example is successful retry behavior with rising effective load. Seen locally, retries appear to improve robustness and successful completion. Seen structurally, they may amplify hidden workload, distort capacity planning, and increase cost without increasing true output. A fourth example is the coexistence of strong benchmark results with deployment drift [3]. The benchmark appears to certify competence, while the deployment context reveals instability or mismatch. In each of these cases, the paradox is not a contradiction in the system itself. It is a contradiction in the explanatory frame used to read it.

These examples clarify why SORT-AI becomes most informative in the presence of structural paradox. The framework is not strongest where local diagnostics already yield a complete explanation. It is strongest where those diagnostics remain locally valid yet globally incomplete. Structural paradox is therefore not marginal to the framework. It is one of the clearest practical signals that the framework is being applied at the right level.

9. Mathematical Basis and Domain Mapping of SORT-AI

The present section provides the minimum mathematical basis required to read SORT-AI as a self-contained domain architecture. It is best understood as a minimal formal basis for domain mapping: it supplies the formal interface needed to map AI-domain structures into SORT notation, and it deliberately does not reproduce the full derivation of the SORT operator system. The reader does not need to reconstruct the underlying whitepaper sequence in order to follow the diagnostic logic of this paper; the formal objects defined here are sufficient for AI-domain interpretation. In this sense, the section is self-contained at the AI-domain interpretation level, while the deeper algebraic derivation and calibration remain in the canonical SORT whitepaper line and are referenced rather than restated.

This distinction matters for the role of the present manuscript. A canonical domain paper must be mathematically intelligible enough that a reviewer can understand how structural diagnosis is grounded, what kinds of mathematical objects are being used, and how those objects are mapped onto the AI domain. At the same time, the paper must remain a domain paper rather than becoming a separate mathematics manuscript. The objective is therefore not completeness of derivation, but sufficiency of formal structure for subsequent AI-domain work.

The mathematical architecture introduced below serves four functions. First, it defines the formal objects through which structural states are represented. Second, it specifies the admissible relations among those objects. Third, it establishes how advanced AI systems are read as structured execution

 doi:10.20944/preprints202512.1334.v4

states and operator chains rather than as internal parameter spaces. Fourth, it connects these formal objects to the diagnostic grammar of V1 to V4. The result is a mathematical basis that remains compact enough for the purposes of the present paper while robust enough to support subsequent application-level studies and domain-specific extensions across the AI domain.

9.1. Purpose and Scope of This Section

The purpose of this section is to provide a mathematically explicit basis for the diagnostic logic of SORT-AI. The formalism presented here is sufficient to explain how structural readings are constructed, how admissible compositions are defined, how projection is carried out, and how structural deviation is interpreted in the AI domain. The section therefore establishes the minimum formal substrate required for the present manuscript to stand as a canonical domain reference rather than as a purely conceptual or taxonomic document.

The scope of the section is deliberately limited. Full derivations of the operator algebra, Jacobi consistency, kernel calibration, and validation tolerances are documented in the canonical SORT whitepaper line [41]. Those materials define the deeper mathematical archive to which the present paper remains formally subordinate. What is reproduced here is the minimum structure required to interpret AI systems diagnostically through SORT-AI.

This scope limitation is not a reduction of mathematical seriousness. It is a division of labor between framework derivation and domain interpretation. The current paper introduces no new operators, no new kernel family, and no additional algebraic regime. Its contribution is to show how the existing formal substrate becomes meaningful in the AI domain and how that meaning is organized through the architectural and diagnostic structure developed in the surrounding sections.

9.2. Core Structural Objects

The formal basis of SORT-AI begins with a closed set of structural objects that remain invariant across domains. These objects are not introduced ad hoc for AI, but are inherited from the canonical SORT whitepaper line [41] and interpreted here in the specific context of advanced AI systems. The first object is a closed set of twenty-two idempotent resonance operators. The second is a global consistency projector. The third is a calibrated projection kernel. The fourth is a structured projection space in which all structural readings are performed. Together, these objects define the minimal formal architecture required for structural diagnosis.

The resonance operators provide the elementary structural actions of the framework. They are not model parameters, network weights, or implementation-level control rules. They are formal structural operators that define admissible transformations and compositions within the diagnostic space. The global projector enforces consistency across such compositions and ensures that local structural selections remain anchored to a shared global reading. The kernel introduces the possibility of scale-dependent, non-local structural coupling. The projection space provides the formal environment in which these operations become well-defined.

Formally, the core structural objects are:

- A closed set of 22 idempotent resonance operators $\{\hat{O}_i\}_{i=1}^{22}$.
- A global consistency projector $\hat{H}$.
- A calibrated projection kernel $\kappa(k)$ with reference scale $\sigma_0 = 0.00190643$.
- A structured projection space $\mathcal{H}_R$ in which all SORT readings are performed [42–45].

The importance of these objects in the present manuscript lies not in their abstract existence alone, but in the role they play in making AI systems structurally legible. Later subsections will show how AI execution states, application forms, projection readings, and structural risk fields are defined through these same objects without introducing a separate domain-specific mathematics.

 doi:10.20944/preprints202512.1334.v4

9.3. Algebraic Consistency Conditions

The first formal requirement of the framework is algebraic admissibility. Structural readings are not generated from arbitrary symbolic compositions. They are restricted to a closed algebra of admissible operator relations. This algebraic discipline is what distinguishes the framework from a descriptive metaphor. The mathematical objects used in SORT-AI must compose consistently, remain closed under admissible multiplication, and preserve the consistency conditions inherited from the underlying framework.

Each operator satisfies idempotency:

$$\hat{O}_i^2 = \hat{O}_i, \qquad \forall\, i \in \{1, \ldots, 22\}. \tag{1}$$

Idempotency is important because it encodes structural stability of the operator action. Reapplying the same structural operator does not generate a different action class; it stabilizes the same structural projection. This property is central to the interpretation of the operators as structural filters rather than as iterative numerical transforms.

Operator commutation is closed under the structural algebra:

$$[\hat{O}_i, \hat{O}_j] = \sum_k C_{ij}^k \hat{O}_k, \tag{2}$$

Idempotency and commutator closure together define the admissible operator algebra used for structural readings. The Jacobi identity is satisfied to machine precision, and a global light-balance condition is enforced by a balanced coefficient set. The complete 22×22 commutator structure is documented in the canonical SORT whitepaper line and its archival deposit [41].

These conditions together define an admissible algebra of operator compositions. Not every symbolic operator string constitutes a valid structural reading. Only algebraically admissible compositions are used for diagnosis, and structural adjacency rules determine which operator sequences count as valid transitions. This point is important in the AI domain because later operator chains will be interpreted as structured execution forms. Their admissibility must therefore be guaranteed at the algebraic level before any diagnostic interpretation is attempted.

9.4. The Global Projector $\hat{H}$

The second fundamental object is the global consistency projector $\hat{H}$. Whereas the resonance operators define local structural actions, the global projector enforces consistency across the resulting structural reading. It ensures that admissible local selections remain embedded in a globally coherent projection regime rather than fragmenting into unrelated local views.

The global consistency projector is defined spectrally as a balanced superposition of the operator set and satisfies global idempotency:

$$\hat{H}^2 = \hat{H}. \tag{3}$$

This condition expresses more than formal convenience. It means that the global reading is itself stable under repeated projection. Once the structural state has been rendered into a globally admissible form, reapplication of the same consistency projector does not alter the admissible class of the reading. In this way, $\hat{H}$ plays the role of a structural anchor: it prevents the projection space from decomposing into incompatible local sub-readings.

For the AI domain, this is especially important because local structural selections may correspond to different components, runtime states, or control surfaces. The role of $\hat{H}$ is precisely to ensure that such local distinctions do not eliminate the possibility of a coherent system-level interpretation. The global projector therefore functions as the formal analogue of system-level structural consistency.

 doi:10.20944/preprints202512.1334.v4

9.5. The Projection Kernel $\kappa(k)$

The projection kernel governs how structural relations are transmitted across scale, abstraction, or separation within the projection space. Without such a kernel, the framework would remain limited to strictly local operator compositions. The kernel introduces the possibility of non-local coupling, which is indispensable for any structural reading of advanced AI systems whose relevant behavior often arises across layers, timescales, or infrastructural boundaries.

The kernel governs scale-dependent non-local projection:

$$\kappa(k) = \exp\left[-\frac{(\sigma_0 L k)^2}{2}\right], \tag{4}$$

with reference scale $\sigma_0 = 0.00190643$ and a characteristic scale L interpreted in the appropriate domain context. The value $\sigma_0 = 0.00190643$ is an inherited calibrated scale parameter of the frozen SORT reference line [41]; it is not re-estimated, re-fitted, or re-calibrated from AI-domain data in the present manuscript. For AI-domain use, the kernel form is read on domain-normalized structural modes rather than as a new AI-specific empirical calibration, so no universality claim about σ_0 is introduced here. In the AI domain, L is read as a structural scale parameter rather than as a geometric length; it indexes coupling depth, separation across execution layers, or other domain-relevant structural distances without changing the shared kernel form inherited from the underlying SORT framework. The kernel construction follows standard operator-theoretic patterns [46,47].

The analytical meaning of the kernel depends on the domain. In physical settings it may be associated with scale or spatial separation. In the AI domain, however, its role is structural rather than geometric in the ordinary sense. Here the kernel models non-local coupling across layers, components, control surfaces, deployment stages, or abstraction levels. It provides the formal mechanism by which the framework can represent the fact that the structurally relevant interaction may occur across boundaries that are not adjacent in a narrow implementation sense.

This point is central for the present manuscript. If advanced AI systems are to be read as structurally coupled systems, then the formalism must allow non-local structural influence. The kernel is the object that makes this possible. It is therefore the mathematical bridge between local operator composition and system-level structural diagnosis.

9.6. AI Systems as Structured Execution States

The next step is to define what kind of formal object corresponds to an AI system in the domain interpretation of SORT-AI. The framework does not attempt to reproduce a model's internal parameters, gradient geometry, or hidden-state dynamics directly. Instead, it treats AI systems as structured execution states. This is a deliberate abstraction. The object of diagnosis is not the internal numerical realization of a model, but the structural condition of the system as it executes within a coupled technical environment.

The abstract AI system state is:

$$S_{\mathrm{AI}} \in \mathcal{H}_R, \tag{5}$$

where S_{AI} encodes runtime configuration, orchestration conditions, data-flow context, control surfaces, and tool or agent context. The state does not encode model weights or internal gradients.

This definition makes two commitments clear. First, S_{AI} is architecture-agnostic. The same structured projection space can accommodate inference workloads, training regimes, hybrid control surfaces, and agentic execution patterns. Second, S_{AI} is a structural abstraction rather than a simulation of the implementation. It captures the system as a diagnostically meaningful execution state, not as a reconstruction of all underlying mechanics.

This abstraction is essential for the role of the domain paper. A canonical domain architecture must remain applicable across diverse AI system classes without being tied to one model family, serving stack, or internal representation format. The concept of a structured execution state provides

 doi:10.20944/preprints202512.1334.v4

exactly that degree of formal generality while remaining specific enough to support domain-level diagnosis.

9.7. AI-Specific Operator Chains

Once the AI system is interpreted as a structured execution state, structural composition can be represented through operator chains. These chains capture the fact that advanced AI systems are not static objects but composed sequences of structurally meaningful actions, mediations, and control relations. The application layer of the domain will later be interpreted as identifying recurrent classes of such compositions.

Structural composition in AI systems is represented as an operator chain:

$$\hat{J}_{\mathrm{AI}} = \prod_{m=1}^{M} \hat{O}_{i_m}, \qquad \hat{O}_{i_m} \in \{\hat{O}_i\}_{i=1}^{22}. \tag{6}$$

Here $\hat{J}_{\mathrm{AI}}$ takes values in the operator algebra generated by $\{\hat{O}_i\}_{i=1}^{22}$, rather than in the generator set itself. Its admissibility is governed by the commutator closure relation in Equation (2) and by the structural adjacency rules of the framework. This formalism should not be read as a low-level execution trace. The operator chain is not a literal sequence of implementation steps. It is a structural composition class. Different applications correspond to admissible classes of such chains, and different AI problem forms select different chain families within the same shared algebra. A runtime-control application and an agentic-emergence application may therefore both be represented by operator chains while remaining structurally distinct because they occupy different admissible composition classes.

The operator-chain representation is what allows the domain architecture to remain both unified and differentiated. Unified, because the same formal substrate is used across the AI domain. Differentiated, because specific applications and clusters correspond to different admissible chain structures rather than to different mathematical worlds.

9.8. Kernel-Modulated Diagnostic Projection

The formal object that renders an operator chain diagnostically readable is the kernel-modulated projection. This is the point at which local operator composition, global consistency, and non-local coupling are brought together into a single diagnostic act. The resulting projected quantity is the formal basis of the structural effect space that later sections interpret as V3.

The diagnostic projection reads an operator chain through the kernel-modulated global projector. In Equation (7), the kernel factor $\kappa(k)$ is read as the operator-valued kernel action associated with the relevant structural spectrum, that is, as a spectral weighting rather than as a free scalar inserted into an otherwise unrelated operator product.

$$\hat{P}_{\kappa}(\hat{J}_{\mathrm{AI}}) = \hat{H}\,\kappa(k)\,\hat{J}_{\mathrm{AI}}\,\hat{H}. \tag{7}$$

The structure of this expression is important. The chain $\hat{J}_{\mathrm{AI}}$ represents the admissible structural composition. The kernel $\kappa(k)$ renders non-local structural coupling visible. The projector $\hat{H}$ enforces global consistency before and after the chain is read. The result is not merely a transformed object, but a diagnostic projection in the precise sense that the structural condition of the system becomes readable through the joint action of admissible composition, non-local coupling, and global consistency enforcement.

Stability is defined as invariance under kernel-modulated projection and normalization, following standard projection-operator conventions [43,44]. Incoherence, by contrast, is detected through deviations in the projected quantity. This is the point at which the mathematical structure begins to connect directly to the diagnostic language of the domain. What later appears as coherence, instability,

 doi:10.20944/preprints202512.1334.v4

drift, or structural deviation is already implicit in whether the projected chain remains invariant or departs from invariance under admissible perturbation.

9.9. Structural Deviation and Risk Fields

The final formal object needed for domain diagnosis is a measure of structural deviation. Once a projected chain has been defined, the framework requires a way to compare baseline and perturbed structural states. This comparison is not probabilistic in the ordinary empirical sense. It is structural. The purpose is to determine whether a perturbation alters the projected condition of the system in a way that is diagnostically meaningful.

Structural drift and coherence loss are read as projection deviations:

$$R_{\mathrm{AI}}(\Delta) = \left\| \hat{P}_\kappa(\hat{\jmath}_{\mathrm{AI}}(\Delta)) - \hat{P}_\kappa(\hat{\jmath}_{\mathrm{AI}}(0)) \right\|_{\mathcal{H}_R}. \tag{8}$$

Here $\|\cdot\|_{\mathcal{H}_R}$ denotes the Hilbert–Schmidt norm on the structured projection space $\mathcal{H}_R$, chosen because it provides a basis-independent measure of projection deviation for the operator-valued quantities considered in the present diagnostic setting. Here Δ denotes a structural perturbation of the system state, and R_{AI} denotes a structural risk field rather than an empirical probability. This distinction is essential. The framework does not claim to infer the frequency of failure or to estimate a probabilistic hazard rate in the usual operational sense. It identifies the magnitude of structural deviation induced by a perturbation in the projection space $\mathcal{H}_R$. The object is therefore diagnostic rather than stochastic.

The norm is fixed in $\mathcal{H}_R$, and explicit metric specification prevents norm-consistency gaps. This matters because structural comparison is only meaningful if the underlying comparison space is itself stable. In the context of the present manuscript, R_{AI} is the formal bridge between projected structure and decision relevance. Later sections will interpret this bridge at the level of V4, but the mathematical role is already clear here: structural deviation is the quantity through which stability loss, drift, and coherence failure become comparable within the diagnostic framework.

9.10. Mapping Mathematical Objects to V1 to V4

Table 4. Mapping of SORT-AI mathematical objects to the V1 to V4 diagnostic grammar.

Dimension	Mathematical object	Diagnostic meaning
V1	S_{AI} (Equation (5))	Observed structural pattern in the system state
V2	$\hat{\jmath}_{\mathrm{AI}}$ (Equation (6))	Structural coupling realized through the operator chain
V3	$\hat{P}_\kappa(\hat{\jmath}_{\mathrm{AI}})$ (Equation (7))	Structural effect space read through kernel-modulated projection
V4	$R_{\mathrm{AI}}(\Delta)$ (Equation (8))	Decision-relevant interpretation of projection deviation

The mathematical objects introduced above do not exist alongside the diagnostic grammar as a separate formal track. They are the formal anchoring of that grammar. V1 corresponds to the structured execution state S_{AI} as the locus of the observed condition. V2 corresponds to the operator chain $\hat{\jmath}_{\mathrm{AI}}$ through which structural coupling is realized. V3 corresponds to the projected quantity $\hat{P}_\kappa(\hat{\jmath}_{\mathrm{AI}})$ through which the structural effect space becomes intelligible. V4 corresponds to the deviation field $R_{\mathrm{AI}}(\Delta)$ through which the projected condition becomes decision-relevant.

This mapping is crucial for the internal unity of the framework. It shows that V1 to V4 are not merely textual or conceptual conventions. They are the domain-level interpretive form of formally specified mathematical objects. This allows the framework to remain scientifically coherent across

 doi:10.20944/preprints202512.1334.v4

papers: the architectural language of the domain and the mathematical language of the framework describe the same sequence at different levels of abstraction.

The same mapping also explains how cross-domain continuity is preserved. AI, CX, and QS do not require distinct mathematical foundations in order to maintain distinct domain architectures. What differs across domains is the interpretation of the structured state and the kinds of coupling that dominate its diagnostic reading. The formal relation among state, chain, projection, and deviation remains invariant.

9.11. Incorporating Level-0 Coherence Without Version Dependency

The mathematical basis used here is consistent with the Level-0 coherence axioms of the canonical SORT framework. This point matters because the present paper is not intended to freeze the broader framework into one historical whitepaper version. Instead, it is intended to remain readable and reusable even as deeper framework documents are clarified or extended.

For that reason, the domain paper incorporates Level-0 coherence through consistency rather than through full rederivation. The formal objects and relations used in this section are already sufficient to preserve continuity with the canonical framework. Future revisions that refine Level-0 consistency conditions can therefore be absorbed through citation and formal alignment without altering the architecture of the present manuscript. This protects the role of the paper as a canonical domain reference rather than tying it to a single moment in the evolution of the deeper framework.

The result is a stable relation between framework and domain. The core framework remains the authoritative source for deeper mathematical derivation. The domain paper remains the authoritative source for AI-domain interpretation built on top of that derivation.

9.12. Mathematical Scope of This Paper

The mathematical scope of the present manuscript is deliberately bounded. No new operators, kernels, or consistency conditions are introduced here. The framework remains architecturally frozen at Level-0 in the sense that domain modules do not alter operator weights, kernel calibration, or validation tolerances. All new content in this paper is interpretive and diagnostic rather than mathematical in the core sense.

This boundary should be understood positively. It is what allows the present manuscript to function as a domain architecture paper rather than as a competing framework derivation. The paper establishes the mathematical sufficiency required for AI-domain diagnosis while preserving the role of the canonical framework references as the deeper source of derivation, calibration, and validation.

In practical terms, this means that later AI-domain work can rely on the present section as the mathematical basis for diagnostic reading without having to restate the deeper framework every time. That is precisely the role a canonical domain paper must fulfill.

The present manuscript also fixes a clear role relation between this domain paper and downstream evidence-bearing work. Kernel-damping reconstruction at the application level is documented in the companion Core-3 technical note as an evidence interface for structural risk-transition analysis [19]. The present domain paper defines the architectural and projection-level frame within which such reconstructions become meaningful; the companion technical note provides the reproducible evidence protocol, and its associated reproduction material is reachable through that note rather than cited separately here. MOCK v4 is referenced throughout this manuscript as the frozen structural reference architecture, not as a runtime engine. The evidence material does not define a new MOCK version, and no production validation, runtime optimization, or benchmark claim is made by virtue of its existence.

10. Core Entry Points into the AI Domain

The current public application space of SORT-AI spans 52 applications across five clusters. A canonical domain paper, however, benefits from a smaller set of entry points that make the architecture legible before the full application space is traversed. For that reason, the present manuscript treats three applications as the Core-3 entry points into the AI domain: AI.01, AI.04, and AI.13. These

 doi:10.20944/preprints202512.1334.v4

applications are not elevated because they are the only important applications in the domain, nor because they exhaust the main structural classes of AI systems. They are selected because they represent three complementary structural coupling axes that recur across advanced AI systems: physical and interconnect coupling, logical and runtime-control coupling, and semantic and agentic coupling.

Taken together, the Core-3 define a minimal but representative cross-section of the domain along these three coupling axes. AI.01 anchors the physical/interconnect coupling axis, where system behavior is dominated by physical and topological interdependence across distributed infrastructure. AI.04 anchors the logical/runtime-control coupling axis, where the dominant problem is the coherence of multiple runtime decision layers acting on a shared execution surface. AI.13 anchors the semantic/agentic coupling axis, where amplification, semantic drift, and multi-step interaction produce system behavior that cannot be reduced to the correctness of local components alone. These three applications are therefore not arbitrary examples. They are canonical structural entry points along three structurally distinct coupling axes.

This section introduces the Core-3 at the level required for the domain architecture. Later sections develop AI.04 in greater detail as the canonical worked example of the present paper. The function of the current section is more basic: it establishes why these three applications are privileged as orientation points within the domain, what structural coupling axes they represent, and why they should not be confused with the entirety of the AI application space.

10.1. Why the Core-3 Matter

The Core-3 applications form the primary entry points into the SORT-AI domain because they represent three structurally distinct and highly recurrent regimes: coupling, control, and emergence. These regimes are not only analytically central; they also correspond closely to the pressure points that increasingly dominate advanced AI deployments in practice. Infrastructure-level coupling determines whether large-scale systems can maintain effective capacity under distributed load. Runtime-level control determines whether independently correct decision layers remain globally coherent when composed under scale. Emergence determines whether agentic or recursively structured execution remains stable once non-linear feedback and multi-step amplification become active.

The value of the Core-3 lies in their complementarity. Each one isolates a dominant structural regime while remaining compatible with the same mathematical substrate, the same application logic, and the same V1 to V4 diagnostic grammar. This makes them ideal entry points for a domain paper whose task is not merely to list applications, but to make the architecture itself legible. A reader who understands why AI.01, AI.04, and AI.13 are distinct yet structurally connected has already grasped much of the underlying logic of the AI domain.

At the same time, the Core-3 should not be misunderstood as a closed or exhaustive subset of the domain. They are a canonical starting set, not a reduction of the domain to three cases. Their role is architectural orientation. They define where a reader can begin reading the domain coherently before later sections widen the frame to the broader application space.

Table 5. Core-3 entry points into the SORT-AI domain. Each application represents a distinct structural coupling axis within the AI domain.

ID	Title	Cluster	Structural regime	Coupling axis
AI.01	Interconnect Stability Control	A	Physical and topological coupling	physical / interconnect
AI.04	Runtime Control Coherence	C	Operative control coherence	logical / runtime-control
AI.13	Agentic System Stability	D	Agentic emergence and amplification	semantic / agentic

 doi:10.20944/preprints202512.1334.v4

10.2. AI.01 — Interconnect Stability Control

AI.01 is the canonical entry point along the physical/interconnect coupling axis of the Core-3. It belongs to Cluster A, Coupling, and its structural focus lies in the physical and logical interdependence between accelerators, interconnect fabric, memory paths, and distributed execution topology [1,16,33]. The analytical importance of this application is that it makes visible a class of failures in which component-local adequacy does not imply system-level efficiency. A system may possess nominally sufficient hardware resources and still suffer effective-capacity loss, throughput collapse, or asymmetric scaling behavior because the dominant structural condition lies in the coupling relation among components rather than in the isolated state of the components themselves.

The specific significance of AI.01 for the domain architecture is that it establishes the physical/interconnect coupling side of structural diagnosis. It shows that large-scale AI and HPC systems must often be read through interdependence, topology, and communication constraints rather than through compute abundance alone. This application therefore provides the natural entry into the physical/interconnect coupling regime and serves as the anchor for later infrastructure-related applications in the broader domain.

10.3. AI.04 — Runtime Control Coherence

AI.04 is the canonical entry point along the logical/runtime-control coupling axis of the Core-3. It belongs to Cluster C, Control, and its focus is the incoherence that can arise between schedulers, orchestrators, runtime engines, and policy-enforcement layers when these operate as locally correct but only partially coordinated control structures [17,30,35]. The structural problem is not that a particular controller is defective. It is that the interaction of multiple correct controllers may generate unstable, oscillatory, inefficient, or economically distorted system behavior at the level of the composed runtime.

This application is used as the canonical worked example of the present paper because it captures especially well the central claim of SORT-AI: advanced AI systems often become structurally problematic not because components fail locally, but because correct components lose coherence when composed under scale. Runtime Control Coherence is therefore uniquely suited to demonstrate how the V1 to V4 grammar, the inner scenario-class architecture, and the mathematical basis of the framework connect in practice. Later sections return to AI.04 precisely for that reason.

10.4. AI.13 — Agentic System Stability

AI.13 is the canonical entry point along the semantic/agentic coupling axis of the Core-3. It belongs to Cluster D, Emergence, and its focus is structural amplification, semantic drift, and emergent instability in multi-agent, tool-using, or recursively organized AI workflows [18,31]. The analytical importance of this application lies in the fact that advanced AI systems increasingly include execution structures in which correctness cannot be evaluated stepwise in isolation. Planning, tool invocation, retry logic, delegation, and multi-agent interaction can produce qualitatively new system behavior once composed, even when each step remains locally permissible.

As the semantic/agentic coupling anchor of the Core-3, AI.13 gives the domain architecture an explicit reference for agentic and emergent regimes. It ensures that non-linear amplification is not treated merely as an afterthought to physical or runtime-control coupling, but as a structurally distinct coupling axis with its own diagnostic significance. This is increasingly important as AI systems move from bounded model execution toward more open-ended, multi-step, and recursively mediated interaction patterns.

10.5. Operational Mapping of the Core-3

To make the Core-3 accessible to readers working primarily in ML-systems, distributed-systems, or infrastructure contexts, Table 6 maps each application to the operational surfaces on which its structural condition is typically first observed and to the structural reading SORT-AI adds. The mapping is interpretive: it relates each coupling axis to recognizable engineering surfaces without asserting any benchmark, measurement, or vendor-specific result.

 doi:10.20944/preprints202512.1334.v4

Table 6. Operational mapping of the Core-3 applications. The table relates each coupling axis to the engineering surfaces on which its condition is usually first observed and to the structural reading SORT-AI adds. It is interpretive and does not report measurements.

Application	Operational surfaces where first observed	Structural reading SORT-AI adds
AI.01	Interconnect fabric, collective communication, memory paths, accelerator topology, distributed serving [1,13,33]	Coupling-axis reading of effective-capacity loss arising from interdependence rather than local resource shortfall
AI.04	Schedulers, orchestrators, runtime engines, admission/routing, retry and policy layers [5,15,30]	Cross-layer control-composition reading of incoherence among locally correct controllers
AI.13	Multi-agent workflows, tool invocation, planning loops, verification/execution chains [31,32]	Semantic/agentic reading of amplification and drift across temporally coupled execution steps

10.6. Why the Core-3 Are Not the Whole Domain

Although the Core-3 provide a powerful orientation into the AI domain, they do not constitute the domain as a whole. The full public AI domain spans 52 applications across five clusters, as summarized in Table 3. The Core-3 cover only three structural regimes directly: coupling, control, and emergence. They do not exhaust the learning-related application space of Cluster B, nor the evidence-oriented applications that give Cluster E its growing strategic importance. As a result, they should be read as canonical starting points rather than as a closed summary of the domain.

This distinction matters because the architecture of SORT-AI is intentionally broader than its most visible core papers. Additional applications address benchmark integrity, training-deployment divergence, deployment drift, structural evidence generation, auditability surfaces, and other problem forms that do not reduce neatly to the Core-3. The broader application space developed later in Section 15 makes clear that the domain has already expanded well beyond these initial anchors and is expected to continue doing so through additive extension.

The Core-3 are therefore best understood as architectural orientation devices. They define a coherent starting set through which the reader can grasp the logic of the domain before moving outward into its fuller application space. Their value lies in clarity and representativeness, not in exclusivity.

11. Runtime Control Coherence as Canonical Example

Among the current application space of SORT-AI, AI.04 is the most suitable canonical example for a domain paper because it makes the central logic of the framework visible with unusual clarity. It belongs to Cluster C, Control, yet it also sits at an intersection where coupling, runtime coordination, system-level economics, and evidence relevance converge. This makes it analytically richer than a narrowly bounded control-theory example and more representative of real advanced AI deployments, where schedulers, orchestrators, runtime engines, policy layers, retries, and serving logic interact continuously rather than as isolated subsystems.

The value of AI.04 for the present manuscript lies in the fact that it expresses the core SORT-AI claim in a form that is both technically recognizable and structurally generalizable. The claim is that system-level incoherence can arise even when the relevant local components remain individually correct. Runtime Control Coherence shows this with particular force because its dominant pathology is not local malfunction but globally incoherent composition. This allows the paper to demonstrate how the domain architecture, the V1 to V4 grammar, and the mathematical basis all connect in a single application.

For that reason, AI.04 is used here not as the only important application in the AI domain, but as the principal worked example through which the domain architecture becomes concrete. The role of

 doi:10.20944/preprints202512.1334.v4

the present section is to explain why this application has that status, what paradox it captures, how it is read through V1 to V4, and how it relates to later deepening in the Core Paper series.

11.1. Why AI.04 Is Selected as the Main Example

AI.04 exhibits the prototypical SORT-AI structure in a particularly concentrated form: locally correct components produce globally incoherent behavior when composed under scale [17]. This makes it an especially strong canonical example for a domain paper whose central purpose is to explain why advanced AI systems must be read as coupled structural systems rather than as collections of isolated technical layers. The application does not depend on rare or exotic conditions. It arises naturally wherever multiple control surfaces act on shared execution states with partial visibility and different temporal assumptions.

Runtime Control Coherence also occupies an analytically strategic position inside the AI domain because many concrete AI-infrastructure problems accumulate at the runtime control layer. Throughput collapse, retry amplification, unstable effective capacity, fluctuating latency, and hidden control overhead often appear first as runtime or serving problems even when their deeper causes involve broader coupling or system composition effects [14,15]. As a result, AI.04 is not merely one application among many. It is a point at which multiple structural tensions become visible at once.

A further reason for selecting AI.04 is that it already has three public demonstration scenarios, S1, S2, and S3, which make it possible to show how one application can appear in multiple operational forms without requiring the full mechanism-level depth of the dedicated Core Paper. This makes the application particularly well suited for a domain paper: it is mature enough to be illustrative, structurally rich enough to be representative, and sufficiently reusable to serve as a bridge between the general architecture of the domain and its more specialized papers.

11.2. The Runtime Control Paradox

The central paradox of Runtime Control Coherence is that each control layer in the system may operate correctly in isolation while the composed runtime nevertheless becomes globally incoherent. Schedulers may optimize placement correctly according to their own objective functions. Orchestrators may reschedule or compensate correctly according to deployment logic. Runtime engines may allocate or batch correctly according to execution conditions. Policy layers may enforce valid rules and constraints. Yet when these mechanisms interact across a shared system surface, the resulting behavior may become oscillatory, wasteful, latency-unstable, or economically distorted.

This paradox matters because it breaks a deeply ingrained analytical assumption in systems practice, namely that local correctness should aggregate toward global correctness if all parts are functioning within specification. Runtime Control Coherence shows that this assumption fails once independently correct control loops act on overlapping structural surfaces with only partial coordination. Under scale, the problem is no longer reducible to whether any one controller is correct. The problem is whether the total control stack remains coherent as a composed structure.

The runtime control paradox therefore does not describe a hidden defect in one subsystem. It describes a failure of composition. Oscillation, resource waste, cost escalation, and non-deterministic latency arise *between* correctly operating control loops rather than within any one of them [30,35,40]. This makes AI.04 an exemplary case of structural diagnosis: the phenomenon is real and consequential, yet it becomes analytically legible only when the system is read at the level of coupled control structure rather than isolated controller performance.

11.3. AI.04 Through V1 to V4

The application AI.04 can be read cleanly through the V1 to V4 diagnostic grammar, which is one of the reasons it functions so effectively as a canonical example in this paper.

At the level of **V1**, the observed structural phenomenon is contraction of effective throughput, rising control overhead, growing cost per output, unstable latency, or deteriorating runtime regularity

 doi:10.20944/preprints202512.1334.v4

even while each visible control layer reports nominal operation. The system appears locally healthy and globally stressed at the same time. This is the diagnostic entry point.

At the level of **V2**, the structural cause is circular or mutually destabilizing interference between independently correct control layers operating with partial visibility and different temporal assumptions. The runtime is not governed by one coherent decision process but by several locally rational decision surfaces acting on the same execution environment. The relevant cause is therefore not a controller defect, but a structural coupling regime within the control stack.

At the level of **V3**, the structural effect space is the coherence-informed projection of control interactions. Here the system becomes readable in terms of effective throughput, control overhead, cross-layer coherence, compensatory activity, and related indicators that describe the runtime as a structural control field rather than as a list of independent metrics. This is the level at which SORT-AI provides its distinct analytical contribution, because the runtime condition becomes legible as a coherence problem rather than as a set of disconnected symptoms.

At the level of **V4**, the reading becomes decision-relevant. The point is no longer to optimize each local loop in isolation, but to redesign coordination boundaries, decouple interfering control surfaces, constrain recursive compensations, improve audit surfaces, or alter the architecture of runtime governance itself. The diagnostic sequence therefore ends not in finer local tuning, but in structurally different decisions about how runtime control should be organized.

11.4. Relation to Future Runtime Control Coherence v2

A revised Runtime Control Coherence paper, version 2, is planned as the next major update in the Core Paper series. That future manuscript is expected to deepen the mechanism-level analysis, extend the scenario structure, and refine the treatment of runtime control as a first-order determinant of economic and operational behavior in large-scale AI systems. It will therefore operate at a different level of detail than the present domain paper.

The role of the current manuscript is not to anticipate that full revision, but to provide the stable domain-level frame within which such a revision remains interpretable. The V1 to V4 reading of AI.04 developed here anchors the application conceptually and structurally. It fixes the place of Runtime Control Coherence within Cluster C, clarifies why it is diagnostic rather than merely descriptive, and shows how the application fits into the broader architecture of SORT-AI.

For that reason, the future AI.04 revision is not a prerequisite for the present domain paper, nor does the present paper depend on future elaboration in order to remain coherent. The relation is the reverse: the domain paper provides the canonical architectural and diagnostic frame within which later refinements of AI.04 can be understood as deepening rather than redefining the application.

 doi:10.20944/preprints202512.1334.v4

12. Illustrative Scenario Layer for AI.04

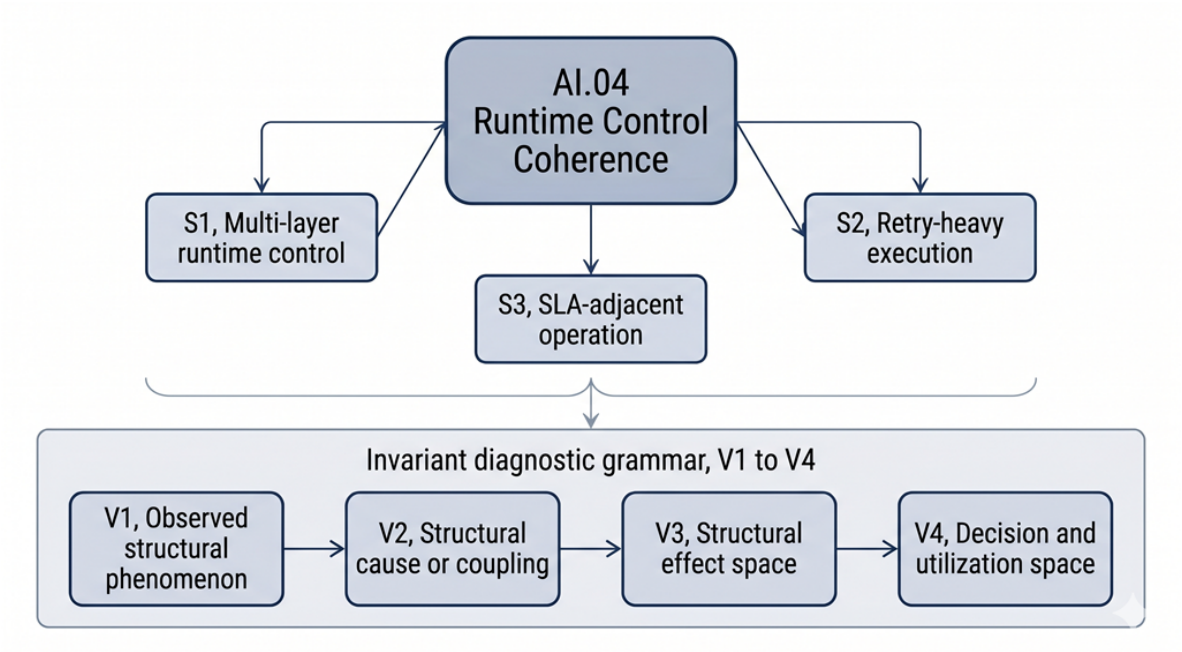

Figure 4. Scenario map for AI.04. The same application diagnoses three distinct operational conditions: multi-layer runtime control, retry-heavy execution, and SLA-adjacent operation. The diagnostic grammar V1 to V4 remains invariant, while the operational surface and dominant control interaction differ across scenarios.

12.1. Purpose of the Scenario Layer

The scenario layer is introduced to show that a single SORT-AI application can appear in multiple concrete operational forms while preserving the same structural logic. This distinction is important for the domain architecture. Applications are recurrent structural problem forms and structured regime spaces; they admit multiple Scenario Classes without fragmenting application identity. A scenario therefore does not define a new application. The three scenarios developed below are early scenario readings of AI.04: scenario-level manifestations within one application form, not separate applications.

For AI.04, three such scenario readings are used: S1, a multi-layer runtime control stack; S2, a retry-heavy execution environment; and S3, SLA-adjacent runtime operation. These scenarios were selected because they span three practically distinct but structurally related forms of control incoherence. In each case, the system remains locally functional and the visible control mechanisms remain individually plausible. Yet in each case the composition of control surfaces produces a runtime condition that cannot be adequately explained through local metrics alone [17].

The scenario layer therefore serves two purposes. First, it demonstrates that AI.04 is not tied to one narrow system class but captures a broader structural control regime. Second, it shows that the V1 to V4 grammar remains invariant even when the operational expression of the problem changes. This is a critical property of a canonical domain paper: the application must remain structurally identifiable across multiple realizations without fragmenting into a set of disconnected case descriptions. A reproducible scenario-class evidence protocol that exercises this inner structure for the Core-3 applications, with declared metric sets and regime classification, is documented separately in the companion Core-3 technical note [19].

12.2. Scenario S1 — Multi-Layer Runtime Control Stack

Scenario S1 represents a deep runtime control stack in which multiple autonomous or semi-autonomous control layers act on a shared execution surface. The system class includes a scheduler, an orchestrator, a runtime engine, and one or more policy-enforcement layers. Each layer is assumed to be locally correct in the sense that it optimizes according to its own objective function and visibility

 doi:10.20944/preprints202512.1334.v4

horizon. The structural problem arises because these objectives and horizons do not align globally, even though none of the participating layers is itself defective.

The observed pattern in S1 is circular interference. Scheduler decisions trigger orchestrator compensations; those compensations alter the runtime assumptions under which the original scheduling decision was locally valid; policy constraints then reshape the admissible execution envelope; and the resulting state feeds back into the next scheduling cycle. The system therefore enters a regime in which locally rational corrections recursively invalidate one another. From a component perspective, each control layer remains within its intended operational logic. From a structural perspective, the control stack becomes incoherent.

The importance of S1 lies in its clarity. It shows AI.04 in its most explicit form: cross-layer decision conflicts become the dominant runtime condition. In this scenario, baseline behavior is structurally incoherent because the control layers are coupled through compensation loops rather than through a coherent coordination boundary. The post-projection interpretation, by contrast, is coherent once cross-layer decision conflicts are explicitly identified and structurally constrained. S1 therefore functions as the cleanest demonstration of Runtime Control Coherence in its pure control-stack form.

12.3. Scenario S2 — Retry-Heavy Execution Environment

Scenario S2 represents an execution environment in which retries are distributed across several layers of the runtime, including scheduler logic, runtime-level recovery, and policy-layer fallback behavior. The system class is not unusual in practice. Retries are often introduced to increase robustness against transient failure, soft timeout, resource contention, or inconsistent downstream response. The structural difficulty arises when these retries remain locally justified while their aggregate effect becomes systemically amplifying.

The observed pattern in S2 is that successful retries conceal growing resource amplification. Completion rates remain high, visible success metrics appear stable, and local failure counts may even look acceptable. Yet the effective cost per successful output rises because the same task may be attempted multiple times across overlapping retry surfaces before final completion. The result is a structural divergence between visible success and actual execution burden. This is why S2 is especially useful for the present paper: it captures a practically important paradox in which operational success metrics remain healthy while the control structure grows economically incoherent.

The structural reading of S2 treats retry loops as hidden feedback amplifiers. The issue is not that retry logic is inherently undesirable. The issue is that layered retries can interact without a shared coherence constraint, producing compounded execution volume, distorted capacity signals, and hidden cost inflation. This makes S2 the most suitable scenario for the reduced mathematical illustration developed in Section 12.6, because the gap between local correctness and global incoherence can be rendered formally without disclosing implementation-specific machinery.

12.4. Scenario S3 — SLA-Adjacent Runtime Operation

Scenario S3 represents runtime operation near service-level boundaries under autoscaling and policy-driven control. The system class is defined by a deployment that remains close to latency or throughput thresholds at which corrective control actions are frequently triggered. This kind of operating regime is common in production environments where cost efficiency requires narrow provisioning margins and where control systems continuously adjust behavior to prevent explicit SLA violation.

The observed pattern in S3 is oscillatory runtime behavior near the threshold surface. Throughput, latency, and cost do not fail catastrophically. Instead, they oscillate together as multiple control layers react to local deviations from target conditions. Autoscaling, admission control, batching policy, or retry handling may each respond correctly in isolation, yet the combined system exhibits self-sustaining oscillation because each intervention alters the conditions assumed by the others. The runtime therefore remains nominally operational while structurally losing coherence.

 doi:10.20944/preprints202512.1334.v4

The structural reading of S3 interprets this behavior as a threshold-coupled control regime without sufficient coherence constraints. What appears at first as high sensitivity or minor runtime volatility is, at the structural level, a regime in which policy and runtime loops interact too close to the decision boundary to remain globally stable. S3 therefore complements S1 and S2 by showing that control incoherence can appear not only through circular interference or hidden amplification, but also through sustained threshold oscillation in environments that remain formally within service bounds.

12.5. What the AI.04 Scenarios Demonstrate

Taken together, the three scenarios demonstrate that a single SORT-AI application can instantiate multiple concrete structural conditions without losing its identity as one application. S1 emphasizes circular interference across a deep control stack. S2 emphasizes hidden amplification through layered retry behavior. S3 emphasizes oscillatory instability near SLA thresholds. The operational surfaces differ, but the structural problem form remains the same: the runtime loses coherence because multiple locally valid control processes interact without preserving a globally coherent execution regime.

This is why the scenario layer matters for the domain architecture. It shows that applications are not reducible to one canonical operational picture. They are structurally recurrent forms that can be realized differently under different technical conditions. The scenario layer provides that variability without dissolving the application into unrelated examples. The underlying V1 to V4 grammar remains stable across instantiations: each scenario exhibits an observed phenomenon, a structural coupling, an effect space in which the condition becomes intelligible, and a decision surface that becomes newly visible through diagnosis.

The scenarios therefore are not separate applications. They are scenario readings — scenario-level manifestations of one application form within the inner regime space of AI.04. This distinction preserves architectural stability while allowing the domain to remain operationally expressive. It also explains why later use cases, demonstrations, and engagement papers can build on the same application while addressing different runtime environments.

12.6. Reduced Mathematical Illustration for S2

A reduced mathematical illustration is introduced for Scenario S2 in order to show that the structural reading of Runtime Control Coherence is formally grounded. Implementation-specific operator selection, kernel parametrization, and scoring functions are outside the scope of this paper. The function of the illustration is to show, in compressed form, how a retry-heavy runtime condition can be represented as a structurally perturbed execution state whose projected coherence differs from the baseline state.

Let

$$\mathbf{x}_{\mathrm{S2}} = \left(\rho_{\mathrm{succ}},\ \mu_{\mathrm{att}},\ c_{\mathrm{succ}},\ \rho_{\mathrm{cas}},\ \alpha_{\mathrm{cost}}\right) \tag{9}$$

denote a normalized indicator vector for Scenario S2, where ρ_{succ} denotes visible success rate, μ_{att} the actual attempt multiplier, c_{succ} the cost per successful completion, ρ_{cas} the retry-cascade frequency, and α_{cost} the cost-attribution accuracy. The precise measurement and weighting of these indicators remain outside the scope of the present manuscript. What matters here is that they define a compact operational state sufficient for a reduced structural reading.

This operational state is mapped into the structured execution state S_{AI} introduced in Equation (5), yielding a scenario-specific chain $\hat{\jmath}_{\mathrm{AI}}^{(\mathrm{S2})}$ and corresponding projected state

$$\hat{P}_{\kappa}\left(\hat{\jmath}_{\mathrm{AI}}^{(\mathrm{S2})}\right) = \hat{H}\,\kappa(k)\,\hat{\jmath}_{\mathrm{AI}}^{(\mathrm{S2})}\,\hat{H}. \tag{10}$$

A baseline retry-heavy regime and a coherence-improved regime can then be compared through the corresponding deviation field

$$\Delta R_{\mathrm{S2}} = \left\| \hat{P}_{\kappa}\left(\hat{\jmath}_{\mathrm{AI}}^{(\mathrm{S2,post})}\right) - \hat{P}_{\kappa}\left(\hat{\jmath}_{\mathrm{AI}}^{(\mathrm{S2,pre})}\right) \right\|_{\mathcal{H}_R}. \tag{11}$$

 doi:10.20944/preprints202512.1334.v4

The interpretation is deliberately qualitative at this stage. In the incoherent baseline regime, the visible success surface ρ_{succ} remains strong while the hidden retry structure amplifies execution burden and distorts cost attribution: μ_{att} and ρ_{cas} rise, c_{succ} grows, and α_{cost} degrades. In the coherence-improved regime, retry interactions are structurally constrained such that visible success and effective execution burden move closer together. The diagnostic significance of Eqs. 9–11 is therefore not that they expose an implementation recipe, but that they show how the paradox of retry-heavy success can be represented formally as a projection difference in the structured space $\mathcal{H}_R$.

To make this concrete, illustrative aggregate values for three principal components of $\mathbf{x}_{\text{S2}}$ are reported in Equation (12). These three components are selected for compact illustration because they capture the dominant visible shift in execution burden, cost formation, and attribution quality, while the remaining components continue to anchor the full scenario vector defined in Equation (9):

$$\left(\mu_{\text{att}},\, c_{\text{succ}},\, \alpha_{\text{cost}}\right)^{(\text{pre})} = (0.58,\, 0.37,\, 0.41) \;\longmapsto\; \left(\mu_{\text{att}},\, c_{\text{succ}},\, \alpha_{\text{cost}}\right)^{(\text{post})} = (0.81,\, 0.14,\, 0.84). \tag{12}$$

These values are stylized demonstrative aggregates intended only to make the projection-difference notation concrete; they do not represent measurement results or a reproducible empirical protocol. The structural meaning is that retry amplification is read as a projection shift in $\mathcal{H}_R$ along the indicator axes that capture hidden execution burden and cost attribution, rather than as a failure of any individual retry mechanism. Specific operator selection within $\hat{\jmath}_{\text{AI}}^{(\text{S2})}$ lies outside the public scope of this paper; the present illustration is intended only to expose the formal diagnostic substrate, not the implementation-specific execution of a full assessment. The underlying operator algebra, kernel construction, calibration logic, and validation structure are documented in the canonical SORT whitepaper line [41].

This reduced illustration is sufficient for the purpose of the present paper. It demonstrates that the scenario layer is mathematically anchored, that structural diagnosis can be expressed formally without turning the domain paper into an implementation document, and that Runtime Control Coherence remains a structural reading rather than a purely verbal reinterpretation of runtime symptoms.

13. How a SORT-AI Application Is Read

The domain architecture developed in the previous sections defines what SORT-AI is, how its application space is organized, and on what mathematical substrate it rests. A canonical domain paper, however, must also explain how the framework is actually used. The present section therefore sets out the reading procedure by which a concrete AI-system condition is translated into a structurally meaningful SORT-AI diagnosis. This procedure is not an implementation workflow, but a diagnostic sequence. Its purpose is to move from visible system behavior to structural interpretation and from structural interpretation to decision relevance.

The procedure is intentionally aligned with the V1 to V4 grammar. This is important because the framework should remain reusable across different clusters, applications, and deployment contexts without requiring a new interpretive method for each case. The reading procedure therefore does not introduce a second layer of logic beyond the architectural structure already established. It operationalizes that structure in a repeatable order. The central premise is that structural diagnosis begins from observable system behavior, but does not end there. It becomes complete only when the observed condition is anchored in a dominant structural regime, matched to a recurrent application form, read through V1 to V4, and translated into a decision surface.

The five steps below should thus be understood as a canonical reading sequence for the domain. In practice, expert diagnosis may move iteratively across them. Yet their logical order remains stable: observation, cluster localization, application identification, diagnostic mapping, and decision relevance. This sequence is one of the reasons SORT-AI can function as a domain architecture rather than as a set of isolated problem descriptions.

 doi:10.20944/preprints202512.1334.v4

13.1. Step 1 — Identify the Observed Phenomenon (V1)

The first step is to identify the observed structural phenomenon. At this stage, the task is to name what appears at the system level without yet attempting to explain it structurally. Typical entry patterns include rising cost per output, unstable latency, oscillatory behavior near service thresholds, throughput contraction, visible drift, degraded reproducibility, capacity loss, or a widening gap between local success metrics and global economic performance. The purpose of this step is descriptive precision. The system-level pattern must be stated clearly enough that it can serve as the entry point for structural diagnosis.

This first step is deliberately non-explanatory. V1 is a description, not yet a diagnosis. It identifies the problem in the form in which it becomes operationally visible. That distinction matters because many analyses of advanced AI systems stop too early at the level of visible symptoms. SORT-AI begins there, but only as a necessary first move. Naming the observed condition correctly prevents the later diagnostic sequence from being built on an ambiguous or excessively local problem statement.

In practical terms, Step 1 asks a simple question: what is actually happening at the level of the system? The answer should be stated in system terms rather than in speculative causal language. If the system shows cost escalation, rising tail latency, soft degradation, retry amplification, or benchmark-deployment divergence, that pattern should be named directly. Only then can the diagnosis proceed toward the structural regime in which the condition becomes intelligible.

13.2. Step 2 — Locate the Dominant Cluster

The second step is to determine the dominant cluster. This is the point at which the observed phenomenon is first placed within the structural architecture of the domain. The task is to decide whether the dominant condition is primarily one of Coupling (A), Learning (B), Control (C), Emergence (D), or Evidence (E). The purpose of the cluster assignment is not to reduce the complexity of the phenomenon artificially, but to identify the structural regime in which the diagnosis should begin.

This step is necessary because the same observed phenomenon may be explainable in more than one way if the structural regime is not fixed. Rising cost, for example, may be a coupling problem, a control problem, or an emergent amplification problem depending on the system class and the form of interaction involved. The dominant cluster fixes the first structural reading and therefore the first plausible application space. It provides the initial interpretive anchor without claiming that the phenomenon is exhausted by a single cluster.

A single phenomenon may indeed span clusters, and later sections of this paper make clear that cross-cluster transitions are often essential to full diagnosis. Nevertheless, the reading procedure requires a dominant entry cluster because the application layer is organized through it. The purpose of Step 2 is therefore not to deny multi-cluster structure, but to prevent diagnostic ambiguity at the point of entry. The cluster assignment establishes the initial structural regime within which the next step becomes possible.

13.3. Step 3 — Identify the Application Form

Once the dominant cluster has been located, the next step is to identify the relevant application form. Here the phenomenon is matched to an existing recurrent structural problem form in the application catalog. The task is not to select a convenient label, but to determine whether the observed condition corresponds to an already established application whose structural logic is sufficient to read the case coherently. This is the point at which the domain architecture becomes operational.

Applications matter because they define recurrent structural forms rather than one-off scenarios. A phenomenon dominated by runtime-level interference between schedulers, orchestrators, and policy layers may map naturally to AI.04. A phenomenon dominated by topology-sensitive effective-capacity loss may map to AI.01. A phenomenon dominated by semantic amplification in multi-agent execution may map to AI.13. The correct application match therefore identifies not merely what the system looks like, but what recurring structural condition it instantiates.

 doi:10.20944/preprints202512.1334.v4

If no existing application fits adequately, the pattern may motivate an additive extension of the application space, as described earlier in Section 6.2. This possibility is important because the application catalog is not treated as closed. At the same time, the threshold for a new application should remain high. Not every unusual deployment condition justifies architectural expansion. A new application is warranted only when the recurring structural form cannot be read coherently through the existing application set.

13.4. Step 4 — Map V1 to V4

Once the application form has been identified, the next step is to fill in the diagnostic grammar V1 to V4. This is where the structural reading becomes explicit. V1 states the observed phenomenon. V2 states the structural cause or coupling that produces it. V3 states the structural effect space through which SORT-AI renders the condition intelligible. V4 states the decision and utilization space that becomes available through that reading. The result is a complete diagnostic sequence that links observation to action without collapsing one into the other.

The analytical center of this step lies in V3 and V4. V1 and V2 are already necessary for entry into the framework, but V3 and V4 are the points at which the specifically SORT-AI reading becomes visible. At V3, the phenomenon is no longer merely described or attributed. It is rendered intelligible as a structural condition inside a coherent diagnostic space. At V4, that structural reading becomes decision-relevant. This is the point at which the framework ceases to be merely classificatory and becomes practically meaningful.

In methodological terms, Step 4 is the completion of diagnosis in the narrow sense. A problem that has been observed, clustered, and matched to an application is not yet fully read until its V1 to V4 structure has been articulated. This is why the diagnostic grammar is central to the canonical domain paper. It provides the invariant interpretive sequence through which applications become scientifically readable across the entire domain.

13.5. Step 5 — Derive Decision Relevance

The final step is to derive decision relevance from the V4 reading. This does not mean that the framework dictates one correct operational intervention. Rather, it means that once the structural condition has been rendered legible, the space of meaningful decisions changes. The system is no longer approached through local symptom treatment alone. Instead, it becomes possible to reason about coordination-boundary redesign, procurement exposure, audit-readiness, architectural decoupling, exit strategies, evidence generation, or governance measures at the level actually appropriate to the diagnosed condition.

This step is the end state of the diagnostic sequence because it completes the transition from structural interpretation to decision surface. A framework that stopped at V3 would remain analytically rich but operationally incomplete. SORT-AI is designed to move one step further. It does not prescribe implementation details, but it clarifies which classes of decision are now structurally relevant and which local interventions are likely to remain insufficient if the coupled condition itself is not addressed.

The importance of Step 5 is especially clear in the broader architecture of this paper. It is the point at which the technical domain can connect to the Sovereign meta-domain, to evidence-oriented deployment reasoning, or to later engagement papers and use cases. Decision relevance is therefore not an auxiliary afterthought. It is the completion of structural diagnosis. The reading sequence ends not when the system has been named correctly, but when the implications of that structural reading have become actionable.

14. Cluster Transitions and Cross-Cluster Phenomena

The cluster architecture of SORT-AI isolates dominant structural regimes, but real AI systems rarely remain confined to a single cluster over the full course of diagnosis. This is not a defect of the architecture. It reflects the fact that advanced AI systems are coupled systems whose relevant behaviors propagate across layers, timescales, and decision surfaces. A phenomenon may therefore

 doi:10.20944/preprints202512.1334.v4

enter diagnosis through one dominant cluster while its downstream consequences become intelligible only through another.

Cross-cluster analysis is necessary because the function of the cluster system is to fix dominant entry regimes, not to imply that real systems decompose into isolated structural classes. A coupling problem may propagate into control incoherence, a control problem may seed emergent amplification, and a learning problem may become operationally visible as a control mismatch. Evidence, finally, intersects all other regimes because structural findings become consequential only when they can be rendered auditable and decision-relevant.

14.1. Why Real Systems Often Cross Clusters

Real AI systems rarely confine a single structural phenomenon to a single cluster because infrastructure coupling, runtime coordination, learning dynamics, emergent behavior, and evidence requirements are typically distributed across the same execution environment. As a result, a condition that first becomes visible as one kind of structural problem may later propagate into another without any sharp ontological break between the two.

This is especially common in large-scale and heterogeneous systems. Interconnect asymmetries may first appear as coupling problems, but later distort scheduler or orchestrator behavior and therefore become control problems. Retry-heavy runtime behavior may first appear as a control problem, but later enter an emergent regime once recursive feedback becomes dominant. Deployment drift may first appear as a learning problem, but become operationally significant only when control policies act on an altered structural state. Cross-cluster reading is therefore often necessary for identifying the true system-level condition.

14.2. Coupling to Control (A to C)

A central transition pattern in advanced AI infrastructure is the shift from Coupling to Control. Here the originating disturbance lies in a physical, logical, or topological coupling surface, but its most relevant downstream effect appears as runtime control incoherence. The system does not cease to have a coupling problem. Rather, that problem becomes operationally legible through the behavior of the control stack. In the inner architecture introduced earlier, such a transition can be read as an overlap regime of the form AI.01 ∩ AI.04: a scenario class in which physical/interconnect coupling and logical/runtime-control coupling interact, and in which infrastructure-side conditions become readable through control-side incoherence.

A canonical example is interconnect asymmetry that induces scheduler or orchestration instability at scale [16,17]. Communication delay, topology-sensitive path variation, or memory-path imbalance may initially appear to be infrastructure-level issues. Yet once these conditions begin to invalidate local control assumptions, scheduler and orchestrator behavior can become compensatory, unstable, or globally inefficient. The result is a coupled A-to-C phenomenon rather than a purely Cluster-A condition. It is important, however, that such an overlap is not automatically a new application. New applications require recurrent metric structure, independent diagnostic relevance, stable V1 to V4 readability, and clear application identity; in the absence of these properties, the transition is documented at the overlap-regime level and AI.01 and AI.04 retain their identities.

14.3. Control to Emergence (C to D)

A second major transition pattern is the shift from Control to Emergence. In this case, the original problem lies in runtime or coordination incoherence, but repeated control interaction produces a regime in which amplification, recursion, or agentic self-reinforcement becomes the dominant structural condition. In the inner architecture introduced earlier, this transition can be read as an overlap regime of the form AI.04 ∩ AI.13: a scenario class in which runtime-control logic interacts with agentic execution and semantic amplification within a single composed runtime.

A clear example is retry amplification escalating into multi-agent or recursive oscillation [31,32]. Retries may initially function as locally reasonable control mechanisms, but once they interact with

 doi:10.20944/preprints202512.1334.v4

planning loops, delegated tasks, tool invocation, or multi-agent verification structures, the resulting behavior may no longer be reducible to a control mismatch alone. The system then enters an emergent regime in which amplification itself becomes the primary structural driver. As before, this overlap is not by itself a new application; it becomes a candidate application only if it shows recurrent metric structure, independent diagnostic relevance, and stable V1 to V4 readability across contexts.

A further admissible overlap regime within the Core-3 is AI.01 ∩ AI.13, in which physical/interconnect constraints interact with agentic orchestration load. This A-to-D transition is not developed in detail here, but it follows the same logic as the A-to-C and C-to-D cases above: it is read at the overlap-regime level, with the existing application identities preserved unless the four candidate-application criteria are satisfied.

14.4. Learning to Control (B to C)

A third recurring transition is the shift from Learning to Control. Here the originating disturbance lies in temporal adaptation, learning drift, or evaluation-deployment divergence, but the dominant operational manifestation appears when control policies continue to act on a system whose structural state has shifted away from the assumptions under which those policies were initially stable.

A representative example is evaluation-deployment divergence that induces policy drift or control mismatch [3]. A model or system may continue to be interpreted through evaluation surfaces that no longer correspond well to deployment behavior. Control policies, routing assumptions, or orchestration rules may then act on an altered state and thereby transform a learning-related discrepancy into a control problem. The consequence is that not all learning-related instability remains confined to Cluster B.

14.5. Evidence as a Cross-Cutting Layer

Cluster E, Evidence, occupies a distinctive place in the transition structure of the domain because it interacts with every other cluster. Evidence is not a separate technical substrate in the sense of coupling, learning, control, or emergence. It is the structural regime in which findings from those clusters become traceable, auditable, and justifiable across the AI stack [21,22].

This cross-cutting role is essential. A coupling diagnosis is incomplete if its implications cannot be rendered visible to architecture and governance decisions. A control diagnosis is incomplete if coherence loss cannot be reconstructed or evidenced. A learning diagnosis is incomplete if deployment drift remains untraceable in a defensible form. An emergence diagnosis is incomplete if recursive amplification cannot be made legible for oversight or audit. For this reason, Evidence should be read as a cross-cutting decision layer rather than as an isolated problem space. This is also why the Sovereign meta-domain depends so strongly on Cluster E.

15. Broader Application Space of SORT-AI

The preceding sections established the canonical structure of the domain through its four axes, cluster architecture, mathematical basis, and Core-3 entry points. A canonical domain paper, however, must also show that the AI domain extends beyond those initial anchors. The purpose of the present section is therefore not to enumerate the full application catalog, but to indicate the broader families of recurrent structural problem forms already represented in SORT-AI.

This broader application space matters because the domain is not reducible to AI.01, AI.04, and AI.13. Those applications provide canonical entry points, but the architecture is designed to absorb additional recurrent problem forms across infrastructure, learning, control, emergence, and evidence without changing its underlying ordering logic. At the same time, the broader application space does not consist only of a list of applications: each application can itself be investigated further through Scenario Classes, Boundary Regimes, and Overlap Regimes, so that the application space has both an extensive dimension (more applications) and an intensive dimension (more structurally legible regimes within each application). The subsections below should therefore be read as structured expansion

 doi:10.20944/preprints202512.1334.v4

rather than as an exhaustive registry, and the inner regime structure introduced in Section 6.4 applies analogously to applications beyond the Core-3.

15.1. Infrastructure and Interconnect Applications

The infrastructure and interconnect region of the application space is associated primarily with Cluster A, Coupling. It includes problem forms in which system behavior is dominated by physical, logical, or topological interdependence across distributed execution surfaces. While AI.01 is the canonical entry point into this regime, the broader family also includes network fabric stability, memory-path coherence, virtualization distortion, heterogeneous accelerator interaction, and deployment-spanning coupling risk.

These applications matter because many contemporary AI systems operate in conditions where effective capacity is determined less by nominal resource abundance than by how coherently those resources remain accessible under coupled runtime conditions [1,13,33]. The unifying principle is that system behavior is shaped by relations among components, paths, and boundaries rather than by the local health of any single element.

15.2. Training, Learning, and Benchmark Applications

The second major region of the broader application space is associated with Cluster B, Learning. Here the dominant structural issue is temporal adaptation. Applications in this family address benchmark integrity, evaluation-context projection instability, objective-constraint divergence, and training-deployment drift [3,34].

What unifies these applications is that the structural condition of the system changes over time, while evaluation and governance surfaces are often comparatively static. The result is a class of problem forms in which systems may remain legible under benchmark or test conditions while drifting away from the conditions under which those evaluations remain operationally meaningful. Cluster B therefore anchors the domain wherever time, adaptation, and shifting interpretive surfaces become structurally central.

15.3. Control and Runtime Applications

The control and runtime region of the domain is associated with Cluster C, Control. While AI.04 is the canonical entry point into this regime, the broader application family includes runtime control coherence, inference pipeline coherence, evaluation-context projection stability, heterogeneous inference-runtime distortion, and structural efficiency recovery [5–7].

These applications are unified by the fact that local correctness at individual control layers does not guarantee coherent global runtime behavior. Throughput contraction, hidden control overhead, latent capacity loss, and policy oscillation often become visible first at the runtime layer even when their upstream causes lie in broader coupling or learning conditions. For this reason, Cluster C acts both as a domain in its own right and as a convergence surface through which deeper structural pressures become operationally visible.

15.4. Emergence and Agentic Applications

The emergent and agentic region of the domain is associated with Cluster D, Emergence. This region addresses problem forms in which the dominant structural issue is non-linear amplification, recursive interaction, semantic drift, or multi-step self-reinforcement. Applications in this family include agentic system stability, multi-agent regime analysis, tool-chain coherence, recursive retry amplification, planning drift, and interaction-induced semantic divergence [31,32].

These conditions are increasingly relevant because advanced AI systems are moving beyond bounded single-pass inference toward architectures involving planning, retrieval, delegation, iterative verification, and cross-agent coordination. Cluster D therefore gives the domain an explicit place for emergent and agentic instability without collapsing such systems back into ordinary runtime-control language.

 doi:10.20944/preprints202512.1334.v4

15.5. Evidence and Assurance Applications

The final region of the broader application space is associated with Cluster E, Evidence. This region addresses structural problem forms in which the central question is whether the condition of an AI system can be made traceable, auditable, reconstructable, and formally justifiable in deployment contexts that require more than operational adequacy.

Applications in this space include structural stability evidence packs, deployment drift signal aggregation, and audit-readiness structures [22,38]. Although the current public application count in Cluster E remains comparatively small, its significance is substantial because it is the layer through which technical findings become institutionally usable. This is also why the Sovereign meta-domain depends so strongly on Cluster E: evidence and assurance applications are the point at which structural diagnosis becomes externally actionable.

This asymmetry reflects publication maturity rather than structural status: Cluster E is expected to expand substantially as evidence-oriented applications accumulate through the additive-extension rule and as assurance-facing deployment contexts become more central to the public research line.

16. How SORT-AI Should Be Read and Used

SORT-AI should be read as a structural interpretation framework for advanced AI systems. Its purpose is not to replace observability, benchmarking, runtime telemetry, or established engineering tools. Its purpose is to make coupled-system conditions readable when local correctness and component-level measurements no longer suffice to explain the behavior of the composed system.

This distinction matters because a domain paper can be misunderstood in two opposite ways. It may be read as if it were proposing a new monitoring or implementation stack, or as if it were only a conceptual taxonomy. Neither reading is adequate. SORT-AI operates at a different level: it provides the architectural and diagnostic grammar through which structurally coupled AI-system conditions become intelligible and decision-relevant.

16.1. SORT-AI as a Structural Reading Framework

SORT-AI is introduced as a structural reading framework rather than as a new measurement system. Its contribution lies in interpretation. The framework does not depend on inventing new basic observables, but on organizing existing system signals through Domain, Cluster, Application, and V1 to V4 so that recurrent structural conditions can be read coherently.

In this sense, SORT-AI functions as a formal diagnostic lens. It identifies when the relevant analytical object is no longer the isolated subsystem but the coupled system as a whole, and it provides a stable language for reading that condition.

16.2. SORT-AI as a Bridge Between Research and Decision

A central feature of SORT-AI is that it connects structural interpretation to decision relevance. This role is carried by the V1 to V4 grammar: V1 and V2 identify the observed condition and its structural cause, while V3 and V4 translate that diagnosis into an effect space and a decision space.

The framework therefore does not stop at description. It renders structural conditions readable in a form that can support architecture, deployment, assurance, audit, procurement, and governance decisions without collapsing those decisions into a single prescribed response.

16.3. SORT-AI as a Complement to Engineering Diagnostics

SORT-AI should be understood as complementary to established engineering diagnostics. Scheduling metrics, runtime telemetry, tracing, benchmarking, and component-level analysis remain indispensable wherever the relevant problem is local. Structural diagnosis becomes necessary where the relevant problem is compositional.

 doi:10.20944/preprints202512.1334.v4

The framework therefore adds an interpretive layer rather than a competing operational stack. It asks whether the coupled system remains structurally coherent under real deployment conditions, even when its components remain locally plausible or locally correct.

16.4. Relation to Existing Analytical Layers

To support external positioning, Table 7 relates SORT-AI to established analytical layers that advanced AI systems already rely on. The intent is not to subsume or replace these layers, but to clarify what SORT-AI adds on top of each. Observability and tracing surface local events and metrics [39]; benchmarking surfaces task and model performance [3]; control theory analyzes specific control loops [48]; governance frameworks address policy, assurance, and accountability [21,22]; and systems and reliability engineering address architecture, deployment, and operational robustness [49–52]. SORT-AI reads across these layers structurally rather than competing with any one of them.

Table 7. Relation between SORT-AI and existing analytical layers. SORT-AI adds a structural reading across coupled regimes; it does not replace the layers listed.

Existing layer	What it primarily observes	What SORT-AI adds
Observability / tracing	Local events, traces, and metrics [39]	A structural reading across coupled regimes
Benchmarking	Task and model performance surfaces [3]	A deployment-structure mismatch reading
Control theory	Specific control loops [48]	A cross-layer control-composition reading
Governance frameworks	Policy, assurance, and accountability [21,22]	A structural evidence interface
Systems / reliability engineering	Architecture, deployment, and reliability practice [49–53]	An application-regime decomposition for AI-system pathologies

The comparison clarifies the analytical level of SORT-AI. Each existing layer remains indispensable within its own scope. SORT-AI becomes useful where a condition is no longer local to one of those layers but distributed across the coupled system, and where reading it requires an application-and-regime decomposition rather than the instrumentation of a single layer.

16.5. Reading Rule for Future Applications

Future applications must remain consistent with the existing cluster architecture, currently defined by Clusters A through E. New applications may extend the domain, but they must enter through a structurally identifiable regime rather than as detached additions.

Application identity is additive and persistent. New applications may supplement the existing set, but they do not remove or redefine established applications. At the same time, V1 to V4 remain the invariant diagnostic grammar across future applications. Their content may vary by case, but their diagnostic role does not.

16.6. Roles of Domain Paper, Technical Note, Application Paper, Evidence Bundle, and MOCK v4

The SORT-AI material around the present domain paper occupies a small number of distinct, complementary roles. Making these roles explicit is part of how SORT-AI should be read and used:

- **Domain Paper.** Defines the architecture and diagnostic grammar of SORT-AI: Domain, Cluster, Application, V1 to V4, and the inner regime structure (Scenario Class, Metric Set, Regime Classification). The present manuscript fulfills this role.
- **Technical Note.** Defines a reproducible evidence protocol for a declared set of applications and scenario classes. The Core-3 kernel-damping technical note is the current example of this role [19].

 doi:10.20944/preprints202512.1334.v4

- **Application Paper.** Provides mechanism-level analysis of a single application, including its specific structural causes, paradoxes, and decision surfaces, while remaining consistent with the architecture defined here.
- **Evidence Bundle.** Is a machine-readable reproduction artefact attached to a technical note: declared inputs, transformation rules, executable scripts, expected and generated outputs, and a reproduction manifest. Evidence bundles do not define a new MOCK version.
- **MOCK v4.** Functions as the frozen structural reference architecture of SORT. It is not an execution engine, not a runtime, and not a benchmark harness; evidence bundles operate on top of MOCK v4 without modifying it.

This role ordering is stated as a positive usage rule. The five artefact classes collaborate, each with a clearly bounded function. A finding becomes architecturally well-formed in SORT-AI when it can be located inside this ordering.

16.7. Scope of Public Scientific Disclosure and Implementation Details

The public scientific scope of this paper consists of the domain architecture, the cluster logic, application identities, V1 to V4 readings, and the mathematical basis developed in Section 9. These elements are sufficient to make the AI domain structurally and mathematically intelligible as a scientific object.

The manuscript does not function as an implementation manual. Implementation-specific details such as scoring functions, internal operator-selection procedures, detailed kernel parametrization strategies, or full assessment execution procedures lie outside its scope. This separation is methodological: the paper provides a complete public domain architecture while leaving implementation-specific execution details outside the present scientific formulation.

17. Strategic Role of the Domain Paper within the Research Program

This manuscript serves as the domain-level reference for SORT-AI. Its role is to provide a stable architectural, diagnostic, and mathematical frame for the AI domain so that later application-specific work can specialize without rederiving the domain structure each time. In this sense, the paper contributes continuity rather than additional application-level depth. The present version extends the architecture by integrating Scenario-Class Architecture and Regime Classification into the application layer while preserving the four main axes—Domain, Cluster, Application, and V1 to V4—unchanged.

17.1. Canonical Reference for the SORT-AI Research Family

The primary function of this paper is to define the common architectural frame of the SORT-AI domain. Subsequent work on specific structural regimes or individual applications can therefore cite the present manuscript for the domain-level logic of Domain, Cluster, Application, and V1 to V4, while remaining focused on mechanism-level or scenario-level analysis.

This role is especially relevant for work centered on the current Core-3, AI.01, AI.04, and AI.13, whose application-specific arguments can be developed in greater technical depth without restating the full AI-domain architecture.

17.2. Relation to Existing Core and Application Papers

The present manuscript replaces the earlier SORT-AI domain paper as the canonical domain reference within the SORT-AI research line. Its purpose is to consolidate the architecture of the domain in a clearer and more self-contained form, not to displace the role of existing Core Papers.

Those papers continue to carry the mechanism-level treatment of their respective applications. The present paper instead provides the domain-level structure within which such application-specific analyses remain mutually interpretable.

 doi:10.20944/preprints202512.1334.v4

17.3. Role in Future Updates

The domain architecture developed here is intended to remain stable as the application space expands. Future applications, scenario layers, and evidence surfaces can be added within the existing structure as long as they remain consistent with the cluster architecture, the additive-extension rule, and the V1 to V4 diagnostic grammar established in this manuscript.

This stability is particularly important for future revisions of existing applications. For example, later work on AI.04 can deepen the runtime-control analysis developed in Sections 11 and 12 without requiring the domain architecture itself to be reformulated. The strategic role of the present paper is therefore to remain the stable reference layer for subsequent development across the SORT-AI domain.

18. Discussion

The present manuscript has introduced SORT-AI as a canonical domain architecture for the structural diagnosis of advanced AI systems. The discussion that follows clarifies the implications of this position at four levels: scientific interpretation, infrastructure relevance, governance and assurance, and the bounded scope of the current domain formulation. The purpose of this section is not to reopen the architecture already established, but to situate it more explicitly within the broader research and deployment landscape in which the paper is intended to operate.

A central result of the preceding sections is that advanced AI systems can no longer be read adequately through isolated model quality, isolated hardware capacity, or isolated runtime metrics alone. The relevant analytical object is increasingly the composed system. Composition does not merely add complexity in a descriptive sense. It generates structural phenomena whose explanatory center lies in coupling, control interaction, emergence, and evidence surfaces. The significance of SORT-AI is that it provides a stable architecture for reading those conditions without collapsing them into generic system difficulty or dispersing them across disconnected case studies.

The discussion therefore emphasizes a positive interpretation of the framework. SORT-AI should be understood neither as a replacement for existing engineering practice nor as a speculative abstraction detached from operational reality. It is a structural diagnostic layer that becomes useful when locally adequate descriptions remain insufficient for understanding the coupled behavior of the whole system. This positioning has consequences for how the framework should be interpreted scientifically, how it can support infrastructure reasoning, how it can connect to assurance and governance, and where its present limits remain.

18.1. Scientific Implications

The first scientific implication of this work is that AI-system phenomena can be formulated as structural problems amenable to operator-based diagnosis. This changes the level of analysis. The framework does not begin with new model dynamics, new empirical variables, or new optimization claims. It begins with the assumption that recurrent AI-system conditions can be rendered intelligible through a stable relation between structured execution states, admissible operator chains, kernel-modulated projection, and decision-oriented diagnostic grammar. In this sense, the paper contributes a structural analysis architecture rather than a new model of AI computation itself.

This point also clarifies the scientific status of the contribution. SORT-AI is diagnostic rather than dynamical. It does not introduce new physical laws, new degrees of freedom, or new empirical parameters. Its purpose is not to compete with machine learning theory, control theory, systems engineering, or infrastructure research on their own internal terms. It provides a Level-0 structural reading of coupled AI systems in which phenomena that otherwise remain scattered across infrastructure, runtime, and deployment literatures can be read within a common architecture. The scientific significance therefore lies in the ordering logic and diagnostic grammar, not in a claim to replace established lower-level theories or methods.

A second scientific implication concerns comparability across problem classes. By organizing the AI domain through Domain, Cluster, Application, and V1 to V4, the framework allows recurrent

 doi:10.20944/preprints202512.1334.v4

conditions to be compared without flattening their differences. Interconnect instability, runtime control incoherence, benchmark-deployment divergence, and agentic amplification can all be read through the same diagnostic grammar while remaining structurally distinct. This is scientifically useful because it enables a domain-wide language of comparison without requiring reduction of all AI-system phenomena to one dominant mechanism.

A third implication concerns the relation between formal and operational reasoning. The mathematical basis introduced in Section 9 does not operate as a decorative formal layer appended to a systems narrative. It anchors the diagnostic sequence formally. This means that the structural language of the domain is not purely metaphorical. State, chain, projection, and deviation are mapped explicitly onto V1 to V4. The scientific significance of this is modest but important: the framework remains formally grounded while staying within the scope of a domain paper rather than becoming a separate mathematics manuscript.

A fourth implication, made explicit by the present version of the manuscript, concerns the inner structure of the application layer. The most important scientific refinement introduced here is that SORT-AI does not merely catalog applications: it treats applications as *structurally assessable regime spaces*. By making Scenario Classes, Metric Sets, and a Core/Boundary/Overlap Regime Classification part of the application layer, the framework supplies a structured vocabulary in which application identity and structural assessment are connected at the same architectural level. This makes the relation between domain architecture and downstream evidence-bearing work explicit rather than implicit.

18.2. Infrastructure Implications

At the level of infrastructure, the present paper provides a shared vocabulary for reading coupling, control, and emergence phenomena in advanced AI systems. This matters because infrastructure teams increasingly encounter conditions that are visible operationally but difficult to classify structurally. Cost escalation, hidden overhead, capacity inaccessibility, throughput contraction, and incoherent scaling behavior are often discussed in separate engineering languages depending on whether they appear in network behavior, runtime coordination, or higher-level orchestration. SORT-AI offers a common structural vocabulary through which such conditions can be interpreted as related but distinct problem forms [14,15].

This vocabulary is compatible with existing infrastructure practice because the framework does not require replacement of schedulers, orchestrators, runtime engines, or serving stacks. It is designed to remain compatible with current systems architectures and current observability practice [5,30]. The value of the framework is not that it introduces a new infrastructure substrate. Its value is that it allows existing infrastructure phenomena to be read at the level of the composed system when local diagnostics cease to be explanatorily sufficient. In this sense, the framework can support infrastructure reasoning without prescribing a uniform operational solution.

A particularly important implication is that infrastructure optimization and structural diagnosis should not be conflated. Existing optimization efforts remain indispensable and often operate correctly within their intended local scope. What SORT-AI adds is the ability to identify when those local optimizations are acting within a coupled system whose dominant problem is no longer local. This can help explain why optimization work sometimes yields diminishing returns despite apparently sound engineering. The issue may not be insufficient local tuning, but an unrecognized structural condition at the level of coupling, control interaction, or emergent composition.

The infrastructure implication is therefore not only conceptual. It concerns decision quality. Once infrastructure teams can distinguish more clearly between coupling-dominated, control-dominated, and emergence-dominated conditions, architectural reasoning becomes more precise. Questions of whether to redesign coordination boundaries, examine topology-sensitive access paths, reinterpret runtime compensation, or reassess agentic orchestration surfaces can be grounded in a clearer structural frame. That is one of the most practical implications of the present paper for large-scale AI system analysis.

 doi:10.20944/preprints202512.1334.v4

The inner regime structure introduced in Section 6.4 reinforces this point. Boundary regimes and overlap regimes are infrastructure- and governance-relevant because they identify where locally plausible systems enter structurally difficult transition states: regions near capacity, control, context, SLA, structural, or validity boundaries, and regions in which two applications interact within a single composed runtime. These are typically the regions in which architectural and oversight decisions become most consequential, and where treating an application as a structured regime space rather than as a flat label is most informative.

18.3. Governance and Assurance Implications

The inclusion of the Sovereign meta-domain earlier in the paper makes clear that structural diagnosis in AI does not end at technical explanation. Through the Sovereign projection layer, SORT-AI connects technical structural findings to governance-relevant decision spaces [22,36,38,54]. This is important because many deployment environments now require more than technical performance. They require auditability, defensibility, traceability, procurement logic, and the ability to justify system-level claims under institutional scrutiny.

Cluster E, Evidence, is central to this connection. The evidence surface of the domain makes it possible to distinguish between a system that functions and a system whose structural condition can be rendered audit-ready. This distinction is likely to become increasingly important in regulated deployments, sovereign infrastructures, critical sectors, and any high-assurance setting in which technical claims must be translated into formal accountability structures. The paper does not provide a full assurance framework in the operational sense, but it establishes the structural conditions under which such assurance becomes thinkable as part of the domain architecture rather than as an external add-on.

A further implication is that governance-relevant interpretation does not require exhaustive disclosure of implementation detail. One of the recurrent tensions in advanced AI deployment is the gap between the need for institutional visibility and the impracticality or inappropriateness of complete implementation exposure. SORT-AI addresses this at the level of structure by distinguishing public scientific architecture from implementation-specific execution details. Evidence surfaces can therefore support audit-readiness and control transparency without requiring the public paper to become an implementation document. This is a methodological contribution, but it is also strategically important for how structural diagnosis may be used in institutional settings.

The broader implication is that governance and assurance should not be treated as purely external overlays on technical systems. Once advanced AI systems are understood as coupled infrastructures with control, emergence, and evidence surfaces, governance becomes structurally internal to how such systems are interpreted. The Sovereign meta-domain does not change the technical substrate. It changes the level at which the technical substrate becomes decision-relevant. This is one of the main reasons the present paper includes it as an early structural element rather than as a late policy appendix.

18.4. Limits and Open Development Paths

The present formulation of SORT-AI is intentionally bounded. It reads coupling, composition, and coherence at the level of structured systems. It does not replace model-internal interpretability methods, representation analysis, alignment-specific mechanism studies, or other approaches whose primary object lies inside model computation rather than in the composed system. This boundary is not a weakness of the domain architecture. It is part of what keeps the framework coherent. The object of SORT-AI is the structural condition of advanced AI systems as composed systems, not the exhaustive analysis of every technically relevant layer.

A second limit is that the domain architecture does not attempt to exhaust the application space in its present public form. The current distribution of 52 applications is substantial, but it is not a terminal catalog. The additive-extension rule explicitly allows the domain to grow when new recurrent structural problem forms emerge. Future work may therefore expand the application space in all five

 doi:10.20944/preprints202512.1334.v4

clusters while preserving the architecture established here. This is especially likely in areas where AI deployment continues to evolve rapidly, such as heterogeneous inference, evaluation instability, agentic coordination, and evidence-oriented deployment reasoning.

A third open path concerns deeper cross-cluster analysis. The present paper has established the principle of cluster transitions and shown why coupling, control, learning, emergence, and evidence cannot always be diagnosed in isolation. Future work can deepen these transitions by formalizing recurrent propagation patterns across clusters and by showing more explicitly how one structural regime seeds another in real systems. This would be especially useful in complex deployment environments where infrastructure, runtime, learning drift, and assurance conditions interact densely.

A fourth open path concerns domain-integrated evidence packs and other structured assurance outputs. The paper has introduced the conceptual and architectural basis for such work, but it does not fully operationalize it. Future research can extend the evidence surface of the domain, strengthen the connection between technical diagnosis and institutional interpretation, and develop more explicit forms of structural justification for high-assurance settings. These developments would remain within the logic of the present architecture rather than requiring a different framework.

The overarching limit, then, is bounded scope. SORT-AI establishes a structural domain architecture for a defined class of AI-system problems and leaves deeper specialization, broader application growth, and richer evidence integration as subsequent development paths. This boundedness is what allows the paper to function as a stable canonical reference rather than as an overextended omnibus manuscript.

19. Conclusions

This manuscript has established SORT-AI as a canonical domain architecture for the structural diagnosis of advanced AI systems. The central result is that advanced AI systems are most productively read not as isolated model artefacts or isolated infrastructure components, but as coupled execution structures whose operationally relevant behavior emerges across layers, boundaries, and control surfaces. In this setting, domain architecture is not secondary to diagnosis. It is the condition under which recurrent structural problem forms become identifiable, comparable, and reusable across research, demonstration, and decision contexts.

The paper has defined this architecture along four axes: Domain, Cluster, Application, and Structural Dimensions V1 to V4. Together, these axes provide the ordering logic through which advanced AI systems become structurally readable. Domain fixes the problem space. Cluster fixes the dominant structural regime. Application fixes the recurrent problem form. V1 to V4 fix the diagnostic grammar linking observed phenomena to structural causes, effect spaces, and decision surfaces. This four-axis architecture is what allows the AI domain to remain stable while the application space can continue to grow additively over time.

19.1. Summary of the Domain Architecture

At the architectural level, SORT-AI has been shown to comprise five clusters, Coupling, Learning, Control, Emergence, and Evidence, spanning 52 current public applications. These clusters separate the dominant structural regimes through which AI-system conditions become diagnostically intelligible while preserving the possibility of cross-cluster propagation and transition. The framework therefore combines stable separation with structural continuity, allowing recurrent problem classes to be distinguished without treating real systems as artificially isolated.

Within this architecture, the Core-3 applications AI.01, AI.04, and AI.13 have been positioned as canonical entry points into the domain. They do not exhaust the domain, but they provide a representative starting set across three highly consequential structural regimes: coupling, control, and emergence. Among them, Runtime Control Coherence, AI.04, has served as the canonical worked example because it makes especially clear how locally correct subsystems can generate globally incoherent behavior when composed under scale.

 doi:10.20944/preprints202512.1334.v4

The paper has also incorporated SORT-Sovereign as a meta-domain layer. This establishes that the significance of structural diagnosis does not end at the technical boundary of AI systems themselves. Structural findings from the AI domain can be projected into strategic, regulatory, procurement, and state-relevant decision spaces without requiring a separate technical formalism. In this way, the domain architecture supports both technical interpretation and institutionally relevant projection.

19.2. Summary of the Mathematical Foundation

To make the domain paper self-contained, a compact mathematical basis has been provided. This basis consists of 22 idempotent operators, a global projector $\hat{H}$, a calibrated kernel $\kappa(k)$, and a structured projection space $\mathcal{H}_R$. These objects define the minimum formal substrate required for structural diagnosis in the AI domain. Their function is not to simulate AI systems at the level of internal implementation detail, but to make structural states, admissible compositions, and projection deviations formally readable.

Within this substrate, AI systems are interpreted as structured execution states, operator chains define recurrent structural compositions, kernel-modulated projection defines the structural effect space, and projection deviation defines structural risk. The paper has further shown that these mathematical objects map directly onto the diagnostic grammar V1 to V4. The consequence is that the architectural language of the domain and the formal language of the framework are not separate layers. They are two levels of articulation of the same structural reading sequence.

The present manuscript has not introduced new operators, kernels, or consistency conditions. Its mathematical contribution is interpretive rather than foundational in the strict sense. It provides the minimum formal basis required for the AI domain to stand independently as a scientific architecture while preserving the deeper framework literature as the authoritative source for full derivation, calibration, and validation.

19.3. Final Positioning

The final position of SORT-AI can therefore be stated precisely. The framework becomes analytically strongest wherever standard diagnostics continue to yield locally plausible or locally consistent findings while the coupled system behaves paradoxically at the system level. Such conditions include local health with escalating cost, local control correctness with global incoherence, successful retries with hidden execution amplification, and acceptable evaluation surfaces with deployment drift. These are not merely practical anomalies. They are structurally meaningful boundary failures and composition failures.

SORT-AI makes such conditions readable through a stable diagnostic grammar and a reusable domain architecture. Its contribution is therefore not to replace engineering diagnostics, benchmarks, or observability systems, but to provide a structural interpretation layer for those AI-system conditions that become decisive only at the level of the composed system. In this sense, the framework remains complementary to established engineering practice while extending the interpretive range available to technical and decision-oriented analysis.

The present manuscript can be summarized in four positive statements. SORT-AI is a Level-0 structural assessment architecture for advanced AI systems. It organizes Applications as recurrent structural problem forms and structured regime spaces. It supports decomposition into Scenario Classes, Metric Sets, and Core, Boundary, and Overlap regimes. It positions SORT-AI as a structural assessment layer above observability, benchmarking, and runtime engineering, without replacing them.

The present manuscript is intended to serve as the canonical reference on which subsequent research and decision-oriented work across the AI domain can be built. Its scientific role is to stabilize the AI domain architecturally. Its strategic role is to preserve continuity across future expansion. As the SORT-AI research line develops, this paper is intended to remain the domain-level reference through which that development remains structurally coherent.

 doi:10.20944/preprints202512.1334.v4

Author Contributions: The author is the sole contributor to this manuscript and was responsible for conceptualization, methodology, formal analysis, manuscript drafting, and final editing.

Funding: This research received no external funding.

Data Availability Statement: The structural and formal basis referenced in this manuscript is archived in public repositories. The canonical SORT framework deposit, which represents the SORT whitepaper line and resolves to its current version, is available via DOI `10.5281/zenodo.17563355`. The public MOCK v4 reference implementation is available via DOI `10.5281/zenodo.18050207`. These archives provide the architectural and formal artefacts that support the framework references used in the present paper. The reproducible evidence material for the SORT-AI Core-3 applications is documented through the companion technical note [19]; the present domain paper does not cite the associated evidence release as a separate primary artefact. As a domain-architecture paper, this manuscript does not claim numerical reproducibility of empirical or system-specific measurements and treats MOCK v4 as the frozen structural reference architecture rather than as a runtime engine. The reproducibility scope is therefore limited to the structural and formal substrate referenced through the archives above.

Acknowledgments: The author acknowledges the use of standard scientific software for document preparation and consistency checking.

Conflicts of Interest: The author declares no conflict of interest.

Use of Artificial Intelligence: Large language models were used as editorial aids for language refinement, structural editing, and LaTeX formatting. All scientific content, including the conceptual structure, mathematical definitions, derivations, diagnostic formulations, and theoretical claims, was produced and verified by the author, who takes full responsibility for the content of this manuscript.

Appendix A. Formula Summary and Notation

This appendix collects the central mathematical objects used throughout the manuscript and cross-references them to the corresponding equations in the main text. No new derivations are introduced here. The appendix serves as a compact reference for notation, formal relations, and their placement within the diagnostic structure of SORT-AI.

Quantity	Equation	Reference
Operator idempotency	$\hat{O}_i^2 = \hat{O}_i$	Equation (1)
Operator commutator closure	$[\hat{O}_i, \hat{O}_j] = \sum_k C_{ij}^k \hat{O}_k$	Equation (2)
Global projector	$\hat{H}^2 = \hat{H}$	Equation (3)
Kernel in Fourier representation	$\kappa(k) = \exp[-(\sigma_0 L k)^2/2]$	Equation (4)
AI system state	$S_{\mathrm{AI}} \in \mathcal{H}_R$	Equation (5)
AI operator chain	$\hat{J}_{\mathrm{AI}} = \prod_{m=1}^{M} \hat{O}_{i_m}$	Equation (6)
Diagnostic projection	$\hat{P}_\kappa(\hat{J}_{\mathrm{AI}}) = \hat{H}\,\kappa(k)\,\hat{J}_{\mathrm{AI}}\,\hat{H}$	Equation (7)
Structural risk field	$R_{\mathrm{AI}}(\Delta) = \|\hat{P}_\kappa(\hat{J}_{\mathrm{AI}}(\Delta)) - \hat{P}_\kappa(\hat{J}_{\mathrm{AI}}(0))\|_{\mathcal{H}_R}$	Equation (8)
S2 indicator vector	$\mathbf{x}_{\mathrm{S2}} = (\rho_{\mathrm{succ}}, \mu_{\mathrm{att}}, c_{\mathrm{succ}}, \rho_{\mathrm{cas}}, \alpha_{\mathrm{cost}})$	Equation (9)
S2 projection difference	$\Delta R_{\mathrm{S2}} = \left\| \hat{P}_\kappa(\hat{J}_{\mathrm{AI}}^{(\mathrm{S2,post})}) - \hat{P}_\kappa(\hat{J}_{\mathrm{AI}}^{(\mathrm{S2,pre})}) \right\|_{\mathcal{H}_R}$	Equation (11)
Illustrative S2 component shift	$(\mu_{\mathrm{att}}, c_{\mathrm{succ}}, \alpha_{\mathrm{cost}})^{(\mathrm{pre})} \mapsto (\mu_{\mathrm{att}}, c_{\mathrm{succ}}, \alpha_{\mathrm{cost}})^{(\mathrm{post})}$	Equation (12)

Notation consistency. All symbols are used consistently throughout the main text and this appendix: $\hat{O}_i$, $\hat{H}$, $\kappa(k)$, $\mathcal{H}_R$, S_{AI}, $\hat{J}_{\mathrm{AI}}$, $\hat{P}_\kappa$, and R_{AI}. Scenario-specific extensions are written in superscript form, for example $S_{\mathrm{AI}}^{(\mathrm{S2})}$, $\hat{J}_{\mathrm{AI}}^{(\mathrm{S2})}$, $\mathbf{x}_{\mathrm{S2}}$, and $(\mu_{\mathrm{att}}, c_{\mathrm{succ}}, \alpha_{\mathrm{cost}})^{(\mathrm{pre})}$, in order to preserve consistency with the internal index structure of the base symbols.

Validation tolerances and algebraic completeness. The validation tolerances associated with the canonical framework, including operator idempotency residuals, commutator residual bounds, Jacobi consis-

 doi:10.20944/preprints202512.1334.v4

tency, light-balance residuals, and the calibrated kernel parameter $\sigma_0 = 0.00190643$, are documented in the canonical SORT framework references. The complete 22×22 commutator structure and the deeper derivational background remain part of the underlying framework literature and are not reproduced in full in the present appendix [41].

Interpretive role of the appendix. The function of this appendix is referential. It provides a compact index of the formal objects that anchor the domain-level diagnostic grammar of SORT-AI. In particular, the relation between structured state, operator chain, diagnostic projection, and structural deviation corresponds directly to the V1 to V4 reading sequence developed in the main text.

References

1. Barroso, L. A.; Hölzle, U.; Ranganathan, P. (2018). *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, 3rd ed. Morgan & Claypool. doi:10.2200/S00874ED3V01Y201809CAC046
2. Dean, J.; Barroso, L. A. (2013). The Tail at Scale. *Communications of the ACM*, **56**(2), 74–80. doi:10.1145/2408776.2408794
3. Kapoor, S.; Widder, D. G.; Ensmenger, N.; Narayanan, A. (2025). Can We Trust AI Benchmarks? An Interdisciplinary Review of Current Issues in AI Evaluation. *arXiv preprint*. arXiv:2502.06559
4. Schaeffer, R.; Miranda, B.; Koyejo, S. (2023). Are Emergent Abilities of Large Language Models a Mirage? *Advances in Neural Information Processing Systems 36 (NeurIPS)*. arXiv:2304.15004
5. Kwon, W.; Li, Z.; Zhuang, S.; et al. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*, 611–626. doi:10.1145/3600006.3613165
6. Patel, P.; Choukse, E.; Zhang, C.; et al. (2024). Splitwise: Efficient Generative LLM Inference Using Phase Splitting. *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*, 118–132. doi:10.1109/ISCA59077.2024.00019
7. Zhong, Y.; Liu, S.; Chen, J.; et al. (2024). DistServe: Disaggregating Prefill and Decoding for Goodput-Optimized Large Language Model Serving. *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. usenix.org/conference/osdi24
8. Yu, G.-I.; Jeong, J. S.; Kim, G.-W.; Kim, S.; Chun, B.-G. (2022). Orca: A Distributed Serving System for Transformer-Based Generative Models. *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 521–538. usenix.org/conference/osdi22
9. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 5998–6008. arXiv:1706.03762
10. Brown, T. B.; Mann, B.; Ryder, N.; et al. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 1877–1901. arXiv:2005.14165
11. Dubey, A.; Jauhri, A.; Pandey, A.; et al. (2024). The Llama 3 Herd of Models. *arXiv preprint*. arXiv:2407.21783
12. Ananthanarayanan, G.; Ghodsi, A.; Shenker, S.; Stoica, I. (2013). Effective Straggler Mitigation: Attack of the Clones. *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 185–198. usenix.org/conference/nsdi13
13. Jeon, M.; Venkataraman, S.; Phanishayee, A.; Qian, J.; Xiao, W.; Yang, F. (2019). Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads. *Proceedings of the USENIX Annual Technical Conference (ATC)*, 947–960. usenix.org/conference/atc19
14. Meta Engineering (2024). Taming Tail Utilization of Ads Inference at Meta Scale. *Meta Engineering Blog*. engineering.fb.com/2024/10/30
15. Databricks Engineering (2024). LLM Inference Performance Engineering: Best Practices. *Databricks Engineering Blog*. databricks.com/blog/llm-inference-performance-engineering
16. Wegener, G. H. (2026). SORT-AI: Interconnect Stability and Cost per Performance in Large-Scale AI Infrastructure. *MDPI Preprints*. doi:10.20944/preprints202601.0161.v1
17. Wegener, G. H. (2026). SORT-AI: Runtime Control Coherence in Large-Scale AI Systems — Structural Causes of Cost, Instability, and Non-Determinism Beyond Interconnect Failures. *MDPI Preprints*. doi:10.20944/preprints202601.0298.v1
18. Wegener, G. H. (2026). SORT-AI: Agentic System Stability in Large-Scale AI Systems. *MDPI Preprints*. doi:10.20944/preprints202601.1741.v1

 doi:10.20944/preprints202512.1334.v4

19. Wegener, G. H. (2026). A Reproducible Kernel-Damping Evidence Protocol for SORT-AI Core-3 Structural Coupling Regimes. *MDPI Preprints*. doi:10.20944/preprints202605.1992.v1
20. Wegener, G. H. (2025). The Supra-Omega Resonance Theory (SORT): A Closed Structural Architecture for Cross-Domain Scientific Analysis. *MDPI Preprints*. doi:10.20944/preprints202511.1783.v3
21. National Institute of Standards and Technology (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1. doi:10.6028/NIST.AI.100-1
22. European Parliament and Council of the European Union (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). *Official Journal of the European Union*, L series, 12 July 2024. eur-lex.europa.eu/eli/reg/2024/1689
23. Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; Mané, D. (2016). Concrete Problems in AI Safety. *arXiv preprint*. arXiv:1606.06565
24. Hendrycks, D.; Carlini, N.; Schulman, J.; Steinhardt, J. (2022). Unsolved Problems in ML Safety. *arXiv preprint*. arXiv:2109.13916
25. Bowman, S. R.; Hyun, J.; Perez, E.; et al. (2022). Measuring Progress on Scalable Oversight for Large Language Models. *arXiv preprint*. arXiv:2211.03540
26. Burns, C.; Izmailov, P.; Kirchner, J. H.; et al. (2023). Weak-to-Strong Generalization: Eliciting Strong Capabilities with Weak Supervision. *arXiv preprint*. arXiv:2312.09390
27. Ji, J.; Qiu, T.; Chen, B.; et al. (2023). AI Alignment: A Comprehensive Survey. *arXiv preprint*. arXiv:2310.19852
28. Koh, P. W.; Sagawa, S.; Marklund, H.; et al. (2021). WILDS: A Benchmark of in-the-Wild Distribution Shifts. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 5637–5664. arXiv:2012.07421
29. Quiñonero-Candela, J.; Sugiyama, M.; Schwaighofer, A.; Lawrence, N. D. (eds.) (2009). *Dataset Shift in Machine Learning*. MIT Press. ISBN 978-0-262-17005-5.
30. Verma, A.; Pedrosa, L.; Korupolu, M.; Oppenheimer, D.; Tune, E.; Wilkes, J. (2015). Large-Scale Cluster Management at Google with Borg. *Proceedings of the Tenth European Conference on Computer Systems (EuroSys)*, Art. 18. doi:10.1145/2741948.2741964
31. Hubinger, E.; van Merwijk, C.; Mikulik, V.; Skalse, J.; Garrabrant, S. (2019). Risks from Learned Optimization in Advanced Machine Learning Systems. *arXiv preprint*. arXiv:1906.01820
32. Greenblatt, R.; Shlegeris, B.; Sachan, K.; Roger, F. (2023). AI Control: Improving Safety Despite Intentional Subversion. *arXiv preprint*. arXiv:2312.06942
33. Jouppi, N. P.; Young, C.; Patil, N.; et al. (2017). In-Datacenter Performance Analysis of a Tensor Processing Unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, 1–12. doi:10.1145/3079856.3080246
34. Maslej, N.; Fattorini, L.; Perrault, R.; et al. (2025). *The AI Index 2025 Annual Report*. Stanford Institute for Human-Centered AI (HAI). aiindex.stanford.edu/report
35. Burns, B.; Grant, B.; Oppenheimer, D.; Brewer, E.; Wilkes, J. (2016). Borg, Omega, and Kubernetes. *Communications of the ACM*, **59**(5), 50–57. doi:10.1145/2890784
36. Shevlane, T.; Farquhar, S.; Garfinkel, B.; et al. (2023). Model Evaluation for Extreme Risks. *arXiv preprint*. arXiv:2305.15324
37. Phuong, M.; Aitchison, M.; Catt, E.; et al. (2024). Evaluating Frontier Models for Dangerous Capabilities. *arXiv preprint*. arXiv:2403.13793
38. National Institute of Standards and Technology (2024). *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*. NIST AI 600-1. doi:10.6028/NIST.AI.600-1
39. Sigelman, B. H.; Barroso, L. A.; Burrows, M.; Stephenson, P.; Plakal, M.; Beaver, D.; Jaspan, S.; Shanbhag, C. (2010). Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. *Google Technical Report*. research.google/pubs/pub36356
40. Saltzer, J. H.; Reed, D. P.; Clark, D. D. (1984). End-to-End Arguments in System Design. *ACM Transactions on Computer Systems*, **2**(4), 277–288. doi:10.1145/357401.357402
41. Wegener, G.H. *Supra–Omega Resonance Theory (SORT): Canonical Whitepaper Line, Versions 3–6 and Later*. Zenodo. DOI: 10.5281/zenodo.17563355.
42. Reed, M.; Simon, B. (1980). *Methods of Modern Mathematical Physics I: Functional Analysis*, revised and enlarged ed. Academic Press. ISBN 978-0-12-585050-6.
43. Halmos, P. R. (1982). *A Hilbert Space Problem Book*, 2nd ed. Springer-Verlag. ISBN 978-0-387-90685-0.
44. Conway, J. B. (2000). *A Course in Operator Theory*. American Mathematical Society. ISBN 978-0-8218-2065-0.
45. Kadison, R. V.; Ringrose, J. R. (1997). *Fundamentals of the Theory of Operator Algebras*. American Mathematical Society. ISBN 978-0-8218-0819-1.

 doi:10.20944/preprints202512.1334.v4

46. Kato, T. (1995). *Perturbation Theory for Linear Operators* (Reprint of 1980 ed.). Springer-Verlag. ISBN 978-3-540-58661-6.
47. Bhatia, R. (1997). *Matrix Analysis*. Springer. ISBN 978-0-387-94846-1.
48. Åström, K. J.; Murray, R. M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press. ISBN 978-0-691-13576-2.
49. Beyer, B.; Jones, C.; Petoff, J.; Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media. ISBN 978-1-491-92912-4.
50. Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media. ISBN 978-1-449-37332-0.
51. Amershi, S.; Begel, A.; Bird, C.; DeLine, R.; Gall, H.; Kamar, E.; Nagappan, N.; Nushi, B.; Zimmermann, T. (2019). Software Engineering for Machine Learning: A Case Study. *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 291–300. doi:10.1109/ICSE-SEIP.2019.00042
52. Paleyes, A.; Urma, R.-G.; Lawrence, N. D. (2022). Challenges in Deploying Machine Learning: A Survey of Case Studies. *ACM Computing Surveys*, **55**(6), 1–29. doi:10.1145/3533378
53. Sculley, D.; Holt, G.; Golovin, D.; Davydov, E.; Phillips, T.; Ebner, D.; Chaudhary, V.; Young, M.; Crespo, J.-F.; Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems. *Advances in Neural Information Processing Systems 28 (NeurIPS)*.
54. Bengio, Y.; Clare, S.; Prunkl, C.; Murray, M.; et al. (2026). *International AI Safety Report 2026*. Independent International Scientific Panel.