Technical Note

A V1–V4 Diagnostic Protocol for SORT-AI Structural Assessment: From AI-Fabric Observation to Scenario-Class Evidence Interfaces

Gregor Herbert Wegener *

Posted Date: 2 June 2026

doi: 10.20944/preprints202606.0148.v1

Keywords: SORT-AI; structural assessment; V1–V4 diagnostic grammar; AI fabrics; scenario classes; metric sets; regime classification; evidence interfaces; kernel-damping compatibility; Level-0 framework

 doi:10.20944/preprints202606.0148.v1

Technical Note

A V1–V4 Diagnostic Protocol for SORT-AI Structural Assessment: From AI-Fabric Observation to Scenario-Class Evidence Interfaces

Gregor Herbert Wegener

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

Abstract

This Technical Note formalizes the public diagnostic protocol by which an observed AI-fabric condition is translated into a SORT-AI structural assessment case. The protocol defines a repeatable path from observation through V_1, V_2, V_3, and V_4 to Application identity, Scenario Class, Metric Set, Regime Classification, and Evidence Interface. It occupies the methodological position between the canonical SORT-AI Domain Paper, which defines the architectural hierarchy and diagnostic grammar, and the Core-3 kernel-damping evidence note, which defines the downstream reproducible risk-transition protocol. The contribution is fivefold: a public assessment grammar; a formal assessment-case object that anchors the diagnostic movement; an Evidence Compatibility Predicate connecting the case to downstream evidence; a public mathematical interface $S_{\mathrm{AI}} \to \hat{J}_{\mathrm{AI}} \to \hat{P}_\kappa(\hat{J}_{\mathrm{AI}}) \to R_{\mathrm{AI}}(\Delta)$; and an explicit public versus implementation-specific boundary. A compact AI.04 illustration instantiates the diagnostic movement at the public grammar level. The note does not claim production validation, benchmark superiority, vendor telemetry analysis, runtime implementation, a new MOCK version, or structural necessity of SORT, and it does not disclose implementation-specific operator selection, telemetry mapping, scoring, weighting, thresholds, intervention playbooks, or production integration architecture.

Keywords: SORT-AI; structural assessment; V1–V4 diagnostic grammar; AI fabrics; scenario classes; metric sets; regime classification; evidence interfaces; kernel-damping compatibility; Level-0 framework

1. Introduction

Advanced AI systems are increasingly deployed as coupled execution fabrics rather than isolated models. In such fabrics, model execution, accelerator allocation, serving logic, scheduling, orchestration, memory management, runtime policy, and evidence surfaces interact across multiple layers. Component-local indicators remain necessary, but they do not determine by themselves whether the composed system remains structurally coherent. This limitation is already visible in large-scale distributed systems, where tail effects, resource sharing, straggler behaviour, and warehouse-scale coordination can dominate aggregate behaviour even when individual components remain within nominal operating bounds [6–10,21].

Recent AI-serving architectures reinforce this point. Paged-attention memory management, distributed serving, prefill/decoding disaggregation, and related inference-system designs show that system-level behaviour depends on interactions among model structure, memory paths, batching, routing, scheduling, and runtime execution constraints [17–20]. Industry-scale engineering reports likewise indicate that tail-utilization effects, inference-performance constraints, and deployment-level execution dynamics cannot be reduced to isolated model capability or accelerator-local metrics alone [13–15]. In parallel, the foundation-model and benchmark literature documents both the rapid

 doi:10.20944/preprints202606.0148.v1

growth of model capability and the limits of model-centric evaluation when deployed behaviour depends on context, orchestration, runtime conditions, and evaluation–deployment mismatch [24–30].

The canonical SORT-AI Domain Paper defines the architectural layer for such analysis: Domain, Cluster, Application, V_1–V_4 diagnostic grammar, Scenario Class, Metric Set, and Regime Classification [1]. The Core-3 kernel-damping evidence note defines a downstream reproducibility protocol for declared structural risk transitions in AI.01, AI.04, and AI.13 [2]. The present Technical Note occupies the methodological position between these two artefacts. It does not restate the SORT-AI domain architecture, and it does not introduce a second evidence protocol. Its purpose is to formalize the diagnostic movement through which an observed AI-fabric condition is translated into a structurally assessable SORT-AI case.

The contribution is a public V_1–V_4 diagnostic protocol that maps an AI-fabric observation to Application identity, Scenario Class, Metric Set, Regime Classification, and Evidence Interface. In compact form, the assessment chain is given in Eq. 10. This chain is the public methodological object of the note. It specifies how a structural diagnosis becomes an assessable case while keeping implementation-specific operator selection, telemetry mapping, scoring functions, weighting logic, intervention playbooks, and production assessment procedures outside the public protocol.

The intended manuscript class is therefore a Technical Note or Methodological Companion Note. The reading order of the three artefacts is

$$\text{Canonical Domain Paper} \rightarrow \text{Present Technical Note} \rightarrow \text{Core-3 Evidence Note.} \tag{1}$$

The first artefact defines the architecture, the present note formalizes the diagnostic protocol, and the third artefact defines the reproducible evidence interface.

1.1. Claim Boundary

This note claims a public diagnostic grammar, an analysis-layer interface, a methodological bridge, a structural assessment path, and an Application-to-Scenario-to-Evidence transition. It claims that SORT-AI Applications can be treated as assessable regime spaces once an observed AI-fabric condition has been ordered through V_1–V_4 and anchored to Scenario Class, Metric Set, and Regime Classification.

The note does not claim production validation, benchmark superiority, vendor telemetry analysis, runtime implementation, a new MOCK version, a production assessment engine, structural necessity of SORT, or disclosure of implementation-specific operator selection, scoring, weighting, telemetry mapping, or intervention logic. The boundary is methodological: the note discloses the public assessment grammar, not the implementation-specific assessment engine.

1.2. Evaluation Frame and Reader Contract

Because the note sits between two SORT-AI artefacts and uses terminology imported from the canonical Domain Paper, its evaluation frame should be explicit. Table 1 states that frame. The purpose is to fix the analytical level of the manuscript for reviewer accessibility while preserving the positive methodological contribution.

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

Evaluate this note as	Do not evaluate it as
A public methodological protocol	A production benchmark
A structural assessment grammar	A runtime implementation
A diagnostic-to-evidence interface	A vendor telemetry study
An analysis-layer formalization	A complete assessment engine
A methodological companion note	A new domain paper

 doi:10.20944/preprints202606.0148.v1

1.3. What is New in This Technical Note

The present note is neither a restatement of the canonical Domain Paper nor a duplicate of the Core-3 Kernel-Damping Evidence Note. It introduces a distinct set of public methodological elements that connect those two artefacts at the analysis layer. Table 2 summarizes these elements and identifies where they are developed in the note.

Table 2. New methodological elements introduced in this Technical Note. Each element is a public analysis-layer construct; none discloses implementation-specific operator selection, telemetry mapping, scoring, weighting, thresholds, or intervention logic.

New element	Contribution
Diagnostics versus Assessment distinction (Section 3)	Separates identifying a structural problem form from making it assessable.
Assessment-case tuple (Section 5)	Defines the public assessment object as a single formal tuple.
Application regime space (Section 6)	Treats Applications as assessable Core, Boundary, and Overlap spaces.
Metric Set and risk transformation layer (Section 7)	Connects Scenario Classes to declared indicators and public transformation roles.
Evidence Compatibility Predicate (Section 8)	Defines when a Scenario Class can connect to the downstream evidence interface.
Public mathematical interface (Section 9)	Separates public assessment reading from implementation-specific execution.

The note can be read independently as a public protocol description: the diagnostic and assessment grammar developed in Sections 4–7 is functionally self-contained at the analysis layer, while the canonical Domain Paper [1] and the Core-3 Evidence Note [2] remain the upstream architectural reference and downstream evidence reference, respectively. The evaluation object of this note is the completeness and traceability of the public assessment grammar.

1.4. Structure of This Note

Section 2 positions the note relative to the canonical SORT-AI Domain Paper and the Core-3 Evidence Note. Section 3 distinguishes structural diagnostics from structural assessment. Section 4 defines the V_1–V_4 diagnostic grammar. Section 5 formalizes the transition from AI-fabric observation to assessment case. Sections 6 and 7 define Applications as assessable regime spaces and specify Scenario Classes, Metric Sets, and Regime Classification. Section 8 describes the Evidence Interface and its compatibility with the existing kernel-damping protocol. Section 9 presents the abstract public mathematical interface. Section 10 gives a compact AI.04 worked mini-example. Section 11 states the public versus implementation-specific boundary. Section 12 discusses the relevance of structural assessment for AI fabrics, Section 13 states the limitations, and Section 14 concludes the note.

2. Position Relative to the Canonical SORT-AI Domain Paper and the Core-3 Evidence Note

The present note is positioned between two SORT-AI artefacts with distinct methodological roles. The canonical SORT-AI Domain Paper defines the architectural hierarchy of the AI domain, including Domain, Cluster, Application, V_1–V_4, Scenario Class, Metric Set, and Regime Classification [1]. The Core-3 Kernel-Damping Evidence Note defines a reproducible downstream evidence protocol for declared structural risk transitions in the Core-3 applications [2]. The present note defines the public transition protocol between these two layers: how an observed AI-fabric condition becomes a structurally assessable case.

The purpose of the present note is therefore narrower: it formalizes how a V_1–V_4 diagnostic reading becomes a structural assessment case. In this sense, it is a Technical Note and Methodological

 doi:10.20944/preprints202606.0148.v1

Companion Note. It is not a second domain paper, because it does not redefine the SORT-AI application catalogue or cluster structure. It is also not a second evidence note, because it does not re-derive the kernel-damping protocol or reproduce the Core-3 calculations. Its role is to define the public diagnostic movement connecting architecture and evidence.

The intended reading order is

$$\text{Canonical SORT-AI Domain Paper} \rightarrow \text{Present Technical Note} \rightarrow \text{Core-3 Evidence Note.} \tag{2}$$

The separation of roles is summarized in Table 3. The Domain Paper supplies the architecture, the present Technical Note supplies the V_1–V_4 assessment protocol, and the Core-3 Evidence Note supplies the downstream reproducibility interface. The three artefacts are therefore sequential but not redundant.

Table 3. Roles of the three artefacts in the present SORT-AI publication sequence. Each artefact has a distinct methodological role and a distinct boundary.

Artefact	Role
Canonical SORT-AI Domain Paper [1]	Defines Domain, Cluster, Application, V1–V4, Scenario Class, Metric Set, and Regime Classification as the canonical Level-0 structural assessment architecture for SORT-AI.
Core-3 Kernel-Damping Evidence Note [2]	Defines a reproducible risk-transition protocol with declared inputs, risk transformations, κ, ξ, CV, classification bands, and reproduction manifest.
Present Technical Note	Defines how a V1–V4 diagnosis becomes a structural assessment case with Application identity, Scenario Class, Metric Set, Regime Classification, and Evidence Interface.

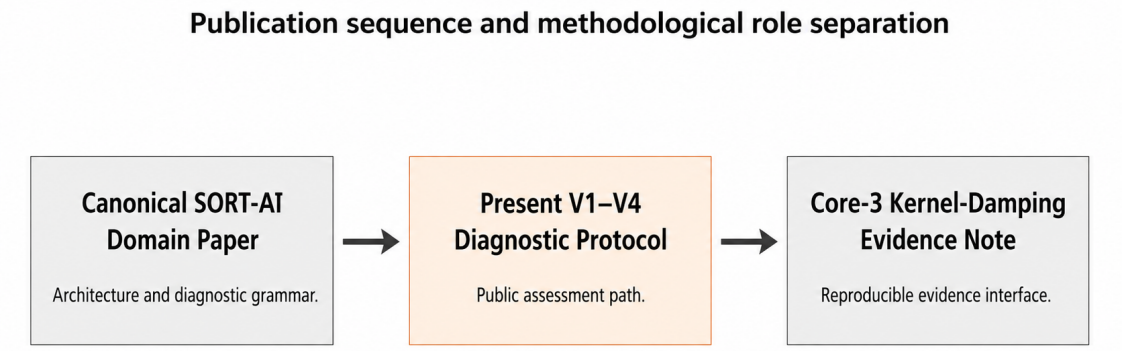

Figure 1. Position of the present Technical Note in the SORT-AI publication sequence. The canonical SORT-AI Domain Paper supplies the architecture and diagnostic grammar; the Core-3 Kernel-Damping Evidence Note supplies the reproducible evidence interface; the present V1–V4 Diagnostic Protocol formalizes the public assessment path that connects them.

2.1. Imported Terms

The note imports several terms from the canonical Domain Paper. Table 4 fixes their meaning within the present protocol so that the assessment grammar can be read without reconstructing the full upstream architecture.

 doi:10.20944/preprints202606.0148.v1

Table 4. Imported SORT-AI terms used in the present note. The right column states the public meaning required to read the note; the canonical Domain Paper [1] provides the full architectural definitions.

Term	Meaning in this note
Domain	AI-fabric problem space under SORT-AI.
Cluster	Structural regime class within the domain.
Application	Recurrent structural problem form, not a software application or deployment-specific use case.
Scenario Class	Typed manifestation inside an Application's regime space.
Metric Set	Declared family of indicators attached to a Scenario Class.
Regime Classification	Assignment of a Scenario Class as Core, Boundary, or Overlap.
Evidence Interface	Compatibility boundary to the downstream reproducibility protocol.
Level-0	Structural assessment layer before implementation-specific telemetry, scoring, and intervention logic.

3. Structural Diagnostics Versus Structural Assessment

Structural diagnostics and structural assessment are sequential but distinct operations. Structural diagnostics identify the problem form within the SORT-AI architecture. Structural assessment then makes that identified condition testable under a declared Scenario Class, Metric Set, Regime Classification, and Evidence Interface.

Diagnostics therefore answers the question: what kind of structural condition is present? Assessment answers the subsequent question: can that condition be made structurally assessable under a declared scenario, metric, and regime? Assessment presupposes diagnosis; without diagnostic anchoring, an assessment case lacks structural identity within the domain architecture. Related asymmetries appear in systems reliability, site reliability engineering, machine-learning evaluation, and reproducibility research, where failure classification, measurement design, and evaluability remain distinct tasks [16,22,23,39].

In SORT-AI, the distinction can be written schematically as

$$\text{Diagnostics} \equiv \text{identification of a structural problem form,} \tag{3}$$

whereas

$$\text{Assessment} \equiv \text{structured evaluation of scenario, metric, and regime behaviour.} \tag{4}$$

The V_1–V_4 diagnostic grammar introduced in Section 4 is the operative bridge between these two levels. It orders the diagnostic movement from observed phenomenon to structural coupling, effect space, and decision surface. Only after this movement has fixed the structural reading can the condition be assigned to an Application identity, Scenario Class, Metric Set, Regime Classification, and Evidence Interface.

4. The V1–V4 Diagnostic Grammar

The V_1–V_4 grammar is the operative diagnostic ordering convention used in this note. It is an epistemic ordering for assessment preparation, not a mechanistic causal chain, not a dynamical model, and not a production diagnostic algorithm. Its function is to order an observed condition so that the condition can be translated into an assessment case [1].

The grammar is invariant in form and domain-specific in content. In the AI domain, V_1 identifies the observed structural phenomenon, V_2 identifies the structural cause or coupling relation, V_3 identifies the structural effect space, and V_4 identifies the decision or utilization surface. The reading order is therefore $V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4$. Distributed tracing and observability systems can expose parts of the V_1 observation surface, but they do not by themselves determine the structural coupling or regime classification [12,16]. Benchmark and model-evaluation literature can contribute to V_3- and V_4-type readings when the assessed object is a model or evaluation regime, but deployment-level

 doi:10.20944/preprints202606.0148.v1

assessment requires an additional structural step from observed behaviour to application-specific regime classification [30–32].

The reading order is

$$V1 \rightarrow V2 \rightarrow V3 \rightarrow V4. \tag{5}$$

Each step contributes a distinct interpretive layer. $V1$ states what is visible. $V2$ identifies the structural relation that makes the observation intelligible. $V3$ locates the resulting condition in an effect space. $V4$ determines what becomes assessable, actionable, or decision-relevant.

Table 5. The V1–V4 diagnostic grammar. Each dimension answers a distinct diagnostic question and contributes a distinct interpretive layer; the four dimensions together order the reading of a structural condition.

Dim.	Object	Diagnostic question	Output
$V1$	Observed structural phenomenon	What is visible at the system level?	Phenomenon statement
$V2$	Structural cause or coupling	What relation produces or organizes the observed condition?	Coupling hypothesis
$V3$	Structural effect space	What structural state class appears once the coupling is read?	Effect-space reading
$V4$	Decision or utilization surface	What becomes assessable, actionable, or decision-relevant?	Assessment class

The four dimensions can be expressed functionally at the public assessment level as

$$V_1(S_{\mathrm{AI}}) = \text{observed phenomenon}, \tag{6}$$

$$V_2(S_{\mathrm{AI}}) = \text{structural coupling relation}, \tag{7}$$

$$V_3(S_{\mathrm{AI}}) = \text{effect-space projection}, \tag{8}$$

and

$$V_4(S_{\mathrm{AI}}) = \text{decision or utilization surface}. \tag{9}$$

Here, S_{AI} denotes the structured AI-system state as observed at the public assessment level. Equations 6–9 do not define an executable transformation. They define the diagnostic roles required before a condition can be assigned to an Application and then refined into Scenario Class, Metric Set, and Regime Classification.

$V1$ is the observation layer. It records the visible system-level condition without yet assigning a structural cause. Examples include rising cost per useful output, degraded effective capacity, unstable tail latency, increasing retry volume, benchmark–deployment divergence, or an auditability gap. $V1$ is therefore not equivalent to raw telemetry; it is the statement of the phenomenon to be assessed.

$V2$ is the structural-coupling layer. It asks what relation organizes the observed condition. In an AI fabric, this relation may involve scheduler–runtime interaction, accelerator placement, memory-path coupling, orchestration feedback, retry logic, policy enforcement, agentic tool-use loops, or evidence-surface fragmentation. $V2$ is the step at which the diagnosis moves beyond local observation toward structural explanation.

$V3$ is the effect-space layer. It identifies the class of structural effects that appears once the coupling relation has been read. Examples include control-coherence loss, retry amplification, interconnect-induced capacity loss, agentic divergence, evaluation–deployment projection mismatch, or evidence incompleteness. $V3$ prepares the transition from diagnostic description to application-level classification.

$V4$ is the decision or utilization surface. It specifies what becomes assessable after the condition has been read structurally. This may include whether the condition is a Core, Boundary, or Overlap regime; whether it requires an evidence interface; whether it affects architectural decisions, governance readiness, or runtime-control interpretation; and whether the condition can be connected to a declared Metric Set for reproducible analysis.

 doi:10.20944/preprints202606.0148.v1

The V1–V4 grammar therefore functions as the operative bridge between structural diagnostics and structural assessment. Diagnostics begins with an observed phenomenon and a structural reading. Assessment begins once that reading can be anchored to Application identity, Scenario Class, Metric Set, Regime Classification, and Evidence Interface.

5. From AI-Fabric Observation to Assessment Case

The V_1–V_4 reading becomes useful for structural assessment only when it is anchored to the domain architecture. The transition from observation to assessment case therefore requires the observed AI-fabric condition to be assigned an Application identity, refined into a Scenario Class, associated with a Metric Set, classified into a Regime, and connected to an Evidence Interface. This transition is the public assessment path formalized in the present note.

This distinction is important because modern AI-fabric observations are distributed across several layers. The relevant observation surface may include distributed traces, service-level signals, runtime logs, model-serving telemetry, memory-management behaviour, scheduler decisions, tool-use traces, and evidence artefacts. Existing observability and site-reliability practices expose parts of this surface [12,16]; production ML and LLM-serving systems further show that model behaviour, runtime infrastructure, and deployment context must be considered jointly [17,20,21]. The present protocol does not replace these observation systems. It specifies how their signals are structurally ordered before assessment.

The full assessment chain is

$$\begin{aligned}\text{Observation} \to V1 \to V2 \to V3 \to V4 \to \text{Application identity} \\ \to \text{Scenario Class} \to \text{Metric Set} \to \text{Regime Classification} \to \text{Evidence Interface}.\end{aligned} \tag{10}$$

Equation 10 is the operational definition of a SORT-AI structural assessment case at the public methodological level. The first segment, from Observation to $V4$, is diagnostic. The second segment, from Application identity to Evidence Interface, is assessment-oriented.

Application identity is fixed during the V1–V4 reading. A Scenario Class does not introduce a new Application; it specifies a typed manifestation of an existing Application under declared structural conditions. A Metric Set then identifies the observable or derived indicators through which the Scenario Class becomes assessable. Regime Classification places the Scenario Class within the internal regime space of the Application, and the Evidence Interface determines whether the resulting case can be connected to a reproducible downstream evidence protocol.

The chain (10) can be summarized as a single formal object. A SORT-AI structural assessment case is the tuple

$$\mathcal{A}_{\text{case}} = \left(S_{\text{AI}},\, V_1, V_2, V_3, V_4,\, A_j,\, C_{j\ell},\, M_{j\ell},\, \rho_{j\ell},\, E_{j\ell}\right). \tag{11}$$

The components in Eq. (11) are declared at the public analysis layer only; the tuple is an interpretive object, not an executable specification. Their public roles are summarized in Table 6.

Table 6. Components of a SORT-AI structural assessment case. Each entry of $\mathcal{A}_{\text{case}}$ is a public analysis-layer object; concrete operator selection, telemetry mapping, and scoring logic are not part of the tuple and remain outside the scope of this note.

Symbol	Public meaning
S_{AI}	Structured AI-fabric state under observation.
V_1, V_2, V_3, V_4	V1–V4 diagnostic reading of S_{AI} (Section 4).
A_j	Application identity within the SORT-AI domain.
$C_{j\ell}$	Scenario Class within Application A_j.
$M_{j\ell}$	Declared Metric Set attached to $C_{j\ell}$.
$\rho_{j\ell}$	Regime classification of $C_{j\ell}$ under $M_{j\ell}$ (Core, Boundary, or Overlap).
$E_{j\ell}$	Evidence Interface (Section 8) to which $C_{j\ell}$ is compatible.

 doi:10.20944/preprints202606.0148.v1

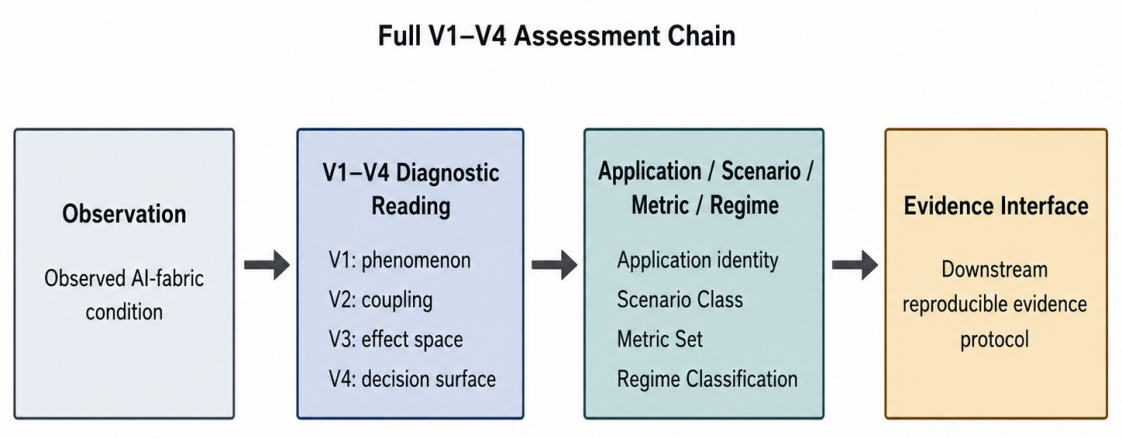

Figure 2. Full V1–V4 assessment chain. An AI-fabric observation is first read through the V1–V4 diagnostic grammar, then anchored to an Application identity, refined into a Scenario Class with declared Metric Set and Regime Classification, and finally connected to a downstream Evidence Interface.

5.1. Notation

The formal objects used in the protocol are collected in Table 7. The table is intended to make the public grammar inspectable as a methodological object. It does not add implementation-specific operator selection, telemetry mapping, scoring, weighting, thresholding, or intervention logic.

Table 7. Consolidated notation used in this Technical Note. Symbols are introduced in the sections indicated and are listed here as a reference.

Symbol	Meaning	Defined in
S_{AI}	Structured AI-fabric state under observation.	Sec. 4, Eq. (11)
V_1, V_2, V_3, V_4	Diagnostic-grammar dimensions: phenomenon, coupling, effect space, decision surface.	Sec. 4, Eqs. (6)–(9)
A_j	Application identity within the SORT-AI domain.	Sec. 6
$C_{j\ell}$	Scenario Class within Application A_j.	Sec. 6, Eq. (14)
$M_{j\ell}$	Declared Metric Set attached to $C_{j\ell}$.	Sec. 7, Eq. (16)
$\rho_{j\ell}$	Regime label of $C_{j\ell}$ (Core, Boundary, Overlap).	Sec. 6, Eq. (15)
$E_{j\ell}$	Evidence Interface attached to $C_{j\ell}$.	Sec. 8
χ_E	Evidence Compatibility Predicate.	Sec. 8, Eq. (20)
κ, ξ, CV	Damping quotient, structure mode, scenario-level coefficient of variation (referenced from [2]).	Sec. 8, Eqs. (22)–(25)
$\hat{J}_{\mathrm{AI}}$	Abstract operator coupling chain (public role only).	Sec. 9, Eq. (26)
$\hat{P}_\kappa$	Kernel-modulated structural projection (public role only).	Sec. 9, Eq. (26)
$R_{\mathrm{AI}}(\Delta)$	Structural deviation or risk field.	Sec. 9, Eq. (26)
$\rightsquigarrow$	Interpretive public assessment reading (not deterministic mapping).	Sec. 9, Eq. (28)

6. Applications as Assessable Regime Spaces

In SORT-AI, an Application is a recurrent structural problem form within a Cluster [1]. It is not a software application, customer deployment, business use case, or isolated incident. The present note treats an Application as an assessable regime space: a structured domain in which Core, Boundary, and Overlap regimes can be distinguished.

This distinction is necessary because an observed condition can belong to the same Application while appearing through different Scenario Classes. A runtime-control condition, for example, may appear as a Core AI.04 condition, as a Boundary condition near an operational threshold, or as an Overlap condition involving infrastructure or agentic coupling. The Application remains the structural problem form; the Scenario Class specifies its assessable manifestation.

 doi:10.20944/preprints202606.0148.v1

The internal regime space of an Application can be written as

$$\mathcal{S}(A_j) \;=\; \mathcal{S}_j^{\text{core}} \cup \mathcal{S}_j^{\text{boundary}} \cup \mathcal{S}_j^{\text{overlap}}. \tag{12}$$

Here $\mathcal{S}(A_j)$ is the public regime space of Application A_j: the family of Scenario Classes available under one Application identity. The three subfamilies are not asserted to be ontologically disjoint; instead, the regime label of a Scenario Class is fixed by the classification function

$$\rho(C_{j\ell}) \;\in\; \{\text{core, boundary, overlap}\}. \tag{13}$$

Core regimes express the central structural mode of the Application. Boundary regimes describe limit cases near capacity, control, context, validity, or evidence boundaries. Overlap regimes identify mixed conditions in which two Applications interact within a single Scenario Class; treating Overlap as a classification output rather than as a hard set partition leaves room for the discovery-signal role discussed below.

This interpretation preserves the stability of the Application layer. Overlap does not automatically imply that a new Application has been discovered. A candidate Application requires recurrence, a stable metric signature, independent diagnostic relevance, and V1–V4 readability. Overlaps are therefore not taxonomy failures. They are discovery signals that may expose cross-application coupling points while leaving the existing Application identity intact.

The Core-3 Applications used throughout this note illustrate the distinction. AI.01 addresses interconnect and infrastructure coupling [3]. AI.04 addresses runtime-control coherence [4]. AI.13 addresses semantic and agentic coupling [5]. These Applications are not use cases. They are structurally distinct problem forms that may each contain Core, Boundary, and Overlap Scenario Classes. Recent work on tool use, reasoning-and-acting workflows, and multi-agent systems motivates the agentic surface on which AI.13 operates, but does not replace the Application-level distinction used here [34–37].

Table 8. Inner structure of an Application. The Application identity is preserved across its Scenario Classes; the Regime Classification situates each Scenario Class within the Application's internal regime space.

Layer	Role
Application identity	Fixed recurrent structural problem form within a Cluster.
Scenario Class	Typed manifestation of the Application under a specific structural condition.
Metric Set	Declared observable or derived indicators through which the Scenario Class becomes assessable.
Regime Classification	Core, Boundary, or Overlap placement within the Application's internal regime space.

7. Scenario Classes, Metric Sets, and Regime Classification

Once an Application has been identified, the assessment case must be refined at the scenario level. A Scenario Class specifies a typed manifestation of the Application. A Metric Set declares the indicator family through which the scenario can be read at the public layer [38,39]. A Regime Classification then assigns the scenario to a Core, Boundary, or Overlap regime, in the sense defined by the classifier of Eq. (15).

This structure prevents two common reductions. A Scenario Class is not a new Application, and a Metric Set is not a complete production telemetry map. The Scenario Class specifies the internal manifestation of an Application, while the Metric Set specifies the public indicator family required to make that manifestation assessable.

Two methodological points follow. First, Overlap regimes are not taxonomy failures: they are discovery signals that may expose cross-application coupling points without by themselves triggering a new Application identity, since the candidate-application criteria of Section 6 continue to apply. Second, Metric-set design is constrained by broader limitations of model and benchmark evaluation

 doi:10.20944/preprints202606.0148.v1

under deployment, which motivates the careful separation between the public indicator family of a Metric Set and any implementation-specific computation of those indicators [23,30].

Table 9. Regime types within the inner regime space of a SORT-AI Application. Counts of typical Scenario Classes are illustrative and depend on the application; see [2] for the Core-3 instantiation.

Regime type	Reading
Core	Central manifestation; expresses the application's axis-defining structural mode.
Boundary	Limit case approaching capacity, control, context, SLA, structural, or validity boundaries.
Overlap	Mixed regime in which two Applications interact within a single Scenario Class.

7.1. Formal Definitions

The verbal definitions above can be stated as analysis-layer relations. None of the following objects discloses operator selection, scoring logic, weighting, telemetry mapping, or production thresholds; each defines only a public structural relation.

Scenario-Class membership.

A Scenario Class is a typed manifestation inside the regime space of its Application:

$$C_{j\ell} \in \mathcal{S}(A_j). \tag{14}$$

Regime classification.

The regime label of a Scenario Class is assigned under its declared Metric Set:

$$\rho_{j\ell} = \rho\Big(C_{j\ell}, M_{j\ell}\Big) \in \{\text{core, boundary, overlap}\}. \tag{15}$$

Eq. (15) treats Core, Boundary, and Overlap as classification outputs rather than as ontological categories, consistent with Section 6. The classifier ρ is treated at the public layer as an axiomatically declared regime-assignment map. Its implementation-specific decision logic is outside the public protocol and is not required for the assessment-case signature. The public role of ρ is to state whether a Scenario Class is read as Core, Boundary, or Overlap within the Application regime space.

Metric Set as declared indicator family.

A Metric Set is a declared family of indicators attached to a Scenario Class:

$$M_{j\ell} = \{m_1, m_2, \ldots, m_n\}_{j\ell}. \tag{16}$$

The members m_i are public or derived indicators; their concrete construction is implementation-specific and is not part of this note.

Metric observation vector.

Evaluating the indicators on the structured AI-fabric state yields the metric observation vector:

$$\mathbf{x}_{j\ell}(S_{\mathrm{AI}}) = \big(m_1(S_{\mathrm{AI}}), m_2(S_{\mathrm{AI}}), \ldots, m_n(S_{\mathrm{AI}})\big). \tag{17}$$

The map $S_{\mathrm{AI}} \mapsto m_i(S_{\mathrm{AI}})$ is declared at the public analysis layer only; this note neither specifies how individual indicators are computed in a deployed system nor how raw telemetry is transformed into S_{AI}.

Public risk transformation.

Each indicator value is converted into a risk representation by a public transformation

$$r_i = T_i(m_i(S_{\mathrm{AI}})), \qquad T_i \in \mathcal{T}_{\mathrm{public}}. \tag{18}$$

 doi:10.20944/preprints202606.0148.v1

The public transformation family contains three role-defining transformations:

$$\mathcal{T}_{\text{public}} = \{T_{\text{risk}}, T_{\text{health}}, T_{\text{overhead}}\}, \qquad T_{\text{risk}}(x) = x, \;\; T_{\text{health}}(x) = 1 - x, \;\; T_{\text{overhead}}(x) = x - 1. \tag{19}$$

The family $\mathcal{T}_{\text{public}}$ defines public transformation roles within the assessment protocol. It does not specify deployment-specific telemetry mappings and does not exhaust the transformations that may be used inside an implementation-specific assessment engine. Its role is limited to the public conversion of an observation into an assessable structural case.

Boundary statement. Metric Sets do not expose telemetry mappings. They declare the public indicator role by which a Scenario Class becomes assessable. The transformation T_i converts an indicator into a risk representation only at the public analysis layer; it does not constitute a scoring function, weighting rule, or threshold definition.

8. Evidence Interfaces and Kernel-Damping Compatibility

The Evidence Interface connects a structurally assessed Scenario Class to a downstream reproducibility protocol [2]. It is not an evidence-generating mechanism inside the present note. Its purpose is to state whether a Scenario Class with a declared Metric Set is compatible with a downstream evidence protocol, such as the Core-3 kernel-damping protocol.

The compatibility relation is expressed by the predicate χ_E. This predicate is syntactic and methodological, not empirical. It states whether the assessment case is sufficiently structured to be passed to the downstream evidence interface; it does not certify production validity, benchmark performance, operational reliability, or empirical sufficiency. Computational reproducibility statements about the evidence protocol belong to [2] and follow established reproducibility, FAIR, software-citation, and artefact-evaluation principles [38–46]; this note neither reproduces them nor extends them.

The verbal compatibility statement above can be expressed as a predicate at the public analysis layer. For a Scenario Class $C_{j\ell}$ with declared Metric Set $M_{j\ell}$, define the Evidence Compatibility Predicate

$$\chi_E(C_{j\ell}) = 1 \iff \forall\, m_i \in M_{j\ell} : \; \big(r_i^{(0)}, r_i^{(1)}\big) \text{ is declared and admissible.} \tag{20}$$

A baseline/comparison risk pair is admissible at the public layer when both entries are declared before interpretation, belong to the same metric family, are finite, and have a non-zero positive baseline wherever a quotient is used. This admissibility condition is a public protocol condition. It does not imply that the selected metrics are empirically sufficient for a production assessment.

The following quantities are used only as interface signatures of the downstream Core-3 protocol. They are not re-derived here and do not define a second evidence method. Their role is to show the type of downstream object to which a compatibility-checked Scenario Class can be connected.

$$r_i^{(0)} \to r_i^{(1)} \qquad \text{(declared risk pair per metric)} \tag{21}$$

$$\kappa_i = \frac{r_i^{(1)}}{r_i^{(0)}} \qquad \text{(damping quotient)} \tag{22}$$

$$\kappa_{\sigma_0}(\xi_i) = \exp\Big[-\tfrac{(\sigma_0 \xi_i)^2}{2}\Big] \qquad \text{(Gaussian kernel form)} \tag{23}$$

$$\xi_i = \frac{\sqrt{-2 \ln \kappa_i}}{\sigma_0} \qquad \text{(implied structure mode)} \tag{24}$$

$$CV_j = \frac{s_{\xi,j}}{\bar{\xi}_j} \qquad \text{(scenario-level coefficient of variation)} \tag{25}$$

 doi:10.20944/preprints202606.0148.v1

The coefficient of variation CV_j is read at the scenario level as a coherence indicator over the implied structure modes associated with a declared Metric Set. In the present note, this reading remains at the public interface level. It does not assert production success, operational improvement, or benchmark superiority.

9. Public Mathematical Interface

The public mathematical interface summarizes the assessment grammar without disclosing implementation-specific operator chains. At the public layer, an AI-system state is represented as a structured input S_{AI}, read through an abstract coupling chain $\hat{J}_{\mathrm{AI}}$, projected through a kernel-modulated public projection $\hat{P}_\kappa$, and interpreted as a structural deviation field $R_{\mathrm{AI}}(\Delta)$.

This interface is intentionally abstract. It is sufficient to state the public assessment grammar, but it is not a customer-specific operator instantiation, telemetry map, scoring function, or intervention procedure.

The public chain is

$$S_{\mathrm{AI}} \longrightarrow \hat{J}_{\mathrm{AI}} \longrightarrow \hat{P}_\kappa(\hat{J}_{\mathrm{AI}}) \longrightarrow R_{\mathrm{AI}}(\Delta). \tag{26}$$

Here, S_{AI} denotes the structured AI-system state at the public assessment level. $\hat{J}_{\mathrm{AI}}$ denotes an abstract operator coupling chain. $\hat{P}_\kappa(\hat{J}_{\mathrm{AI}})$ denotes the kernel-modulated structural projection of that coupling chain. $R_{\mathrm{AI}}(\Delta)$ denotes the resulting structural deviation or risk field. The chain in Eq. 26 is compatible with standard operator-theoretic and matrix-analytic settings [54,55], but the present note does not derive, extend, or depend on those foundations as a new mathematical result.

Table 10. Public mathematical interface objects in the chain of Eq. 26. Each object is declared at the public abstract level; concrete operator selection, telemetry mapping, weighting, scoring, and intervention logic remain outside the scope of this note.

Object	Public meaning
S_{AI}	Structured AI-system state.
$\hat{J}_{\mathrm{AI}}$	Abstract operator coupling chain.
$\hat{P}_\kappa(\hat{J}_{\mathrm{AI}})$	Kernel-modulated structural projection of the abstract coupling chain.
$R_{\mathrm{AI}}(\Delta)$	Structural deviation or risk field.

The interface in Eq. 26 is deliberately abstract. It identifies the public mathematical roles required by the assessment protocol, but it does not disclose the implementation-specific operator instantiation. In particular, the customer-specific form

$$\hat{J}_{\mathrm{AI}}^{\mathrm{customer}} = \hat{O}_i \hat{O}_j \hat{O}_k \cdots \tag{27}$$

is not specified in this note. Equation 27 is shown only to mark the boundary between the public formal interface and a non-disclosed implementation-specific assessment layer. No weighting logic, scoring function, telemetry mapping, production threshold, or intervention rule is defined here.

This boundary is essential for the manuscript's role. The note formalizes a public diagnostic and assessment grammar. It does not publish an operational assessment engine. The same distinction is restated in Section 11, where the public versus implementation-specific boundary is summarized explicitly.

The public chain in Eq. (26) connects to the assessment objects of Sections 5–7 by the public reading

$$R_{\mathrm{AI}}(\Delta) \;\rightsquigarrow\; \left(C_{j\ell},\, M_{j\ell},\, \rho_{j\ell},\, E_{j\ell}\right). \tag{28}$$

The symbol $\rightsquigarrow$ denotes an interpretive public assessment reading. It is not a deterministic inference rule, not an executable mapping, and not a production diagnostic procedure.

 doi:10.20944/preprints202606.0148.v1

10. Worked Mini-Example: AI.04 Runtime Control Coherence

AI.04 Runtime Control Coherence is used as a compact worked mini-example because it illustrates the diagnostic movement from observation to assessability without requiring implementation-specific disclosure [4]. The selected Scenario Class is AI.04.C2, Retry Amplification. The example instantiates the assessment chain at the public grammar level only.

The purpose of the example is to show how an observed runtime-control condition is ordered through V_1–V_4, anchored to an Application identity, refined into a Scenario Class, associated with a declared Metric Set, assigned a Regime Classification, and connected to the Evidence Interface. It does not define a production diagnostic engine. As an alternative overlap reading, the same movement can be read through AI.04.O1, Control plus Infrastructure Coupling. The background is operationally motivated by tail-latency and straggler effects, cross-layer tracing, end-to-end placement of control decisions, and current LLM-serving control surfaces [6,11,12,17,20]. These references motivate the observation surface; they do not define a vendor-specific system or a production measurement claim.

The diagnostic movement is summarized in Table 11. The table should be read as an assessment-chain trace, not as an implementation recipe.

Table 11. AI.04 worked mini-example: assessment-chain trace for AI.04.C2 Retry Amplification. The trace illustrates the public diagnostic movement; no implementation-specific operator selection, telemetry mapping, weighting, scoring, or intervention logic is disclosed.

Step	Reading
$V1$	Rising cost, retry amplification, or reduced effective capacity is observed at the AI-fabric level.
$V2$	The condition is read as scheduler–orchestrator–runtime–retry–policy coupling.
$V3$	The effect space is control-coherence loss, retry amplification, or boundary oscillation.
$V4$	The decision surface concerns boundary redesign, control separation, or evidence readiness.
Application identity	AI.04 Runtime Control Coherence.
Scenario Class	AI.04.C2 Retry Amplification; alternatively AI.04.O1 as an overlap with infrastructure coupling.
Metric Set	Abstract risk, health, or overhead indicators; the set is declared at the assessment level, while implementation-specific contents are not disclosed.
Regime Classification	Core for AI.04.C2; Overlap for AI.04.O1.
Evidence Interface	Risk transition mapped to κ, ξ, and CV under the existing Core-3 evidence protocol [2].

The example shows why the V1–V4 grammar is not merely descriptive. At $V1$, the condition is visible as a system-level phenomenon. At $V2$, the observation is anchored to a structural coupling relation. At $V3$, the condition is placed in an effect space. At $V4$, the condition becomes decision-relevant. Only then can it be assigned to an Application identity, refined into a Scenario Class, associated with a Metric Set, classified as a Core or Overlap regime, and connected to the Evidence Interface.

Specializing the assessment-case tuple of Eq. (11) to this illustration gives

$$\mathcal{A}_{\mathrm{AI.04.C2}} = \left(S_{\mathrm{AI}}, V_1^{\mathrm{AI.04}}, V_2^{\mathrm{AI.04}}, V_3^{\mathrm{AI.04}}, V_4^{\mathrm{AI.04}}, A_{\mathrm{AI.04}}, C_{\mathrm{AI.04.C2}}, M_{\mathrm{AI.04.C2}}, \rho_{\mathrm{core}}, E_{\mathrm{AI.04.C2}}\right). \tag{29}$$

Eq. (29) instantiates the public components only. The metric list inside $M_{\mathrm{AI.04.C2}}$, the operator instantiation underlying $\hat{f}_{\mathrm{AI}}$, and any telemetry mapping that would make $\mathbf{x}_{j\ell}(S_{\mathrm{AI}})$ executable on a deployed system are not disclosed.

The example remains public by construction. It identifies the structural path from observation to assessability, but it does not define a production diagnostic engine. Concrete operator instantiations are governed by the boundary stated in Section 9 and summarized in Section 11.

11. Public Scope and Implementation-Specific Boundary

The present note discloses the public assessment grammar of SORT-AI, not an implementation-specific assessment engine. This boundary is methodological: it identifies the level at which the

 doi:10.20944/preprints202606.0148.v1

protocol can be stated publicly while keeping concrete assessment execution outside the scope of the note.

The public layer consists of the diagnostic and assessment structure developed in the preceding sections: V_1–V_4, Application identity, Scenario Class, Metric Set, Regime Classification, the abstract public mathematical chain in Eq. 26, and the Evidence Interface. This layer makes the method inspectable as a public structural assessment grammar.

The implementation-specific layer consists of operator selection, telemetry mapping, weighting logic, scoring functions, production thresholds, intervention playbooks, and integration architecture. These elements depend on the concrete operational context and are not required to state the public diagnostic protocol. The public versus implementation-specific split is also consistent with AI risk-management and governance practice, where evidence interfaces and traceability requirements can be specified without disclosing the internal implementation logic of the operating organization [47–49].

Table 12. Public scope and implementation-specific boundary of the present Technical Note. The note discloses the public diagnostic and assessment grammar; concrete implementation logic remains outside its scope.

Public in this note	Not disclosed in this note
V1–V4 grammar	Implementation-specific operator chains
Application / Scenario / Metric / Regime hierarchy	Customer telemetry mapping
Abstract risk-transition interface	Scoring functions
Kernel-damping Evidence Interface, referenced only	Weighting logic
AI.04 illustrative assessment path	Production thresholds
Claim boundary	Intervention playbooks
Public mathematical chain in Eq. 26	Production integration architecture

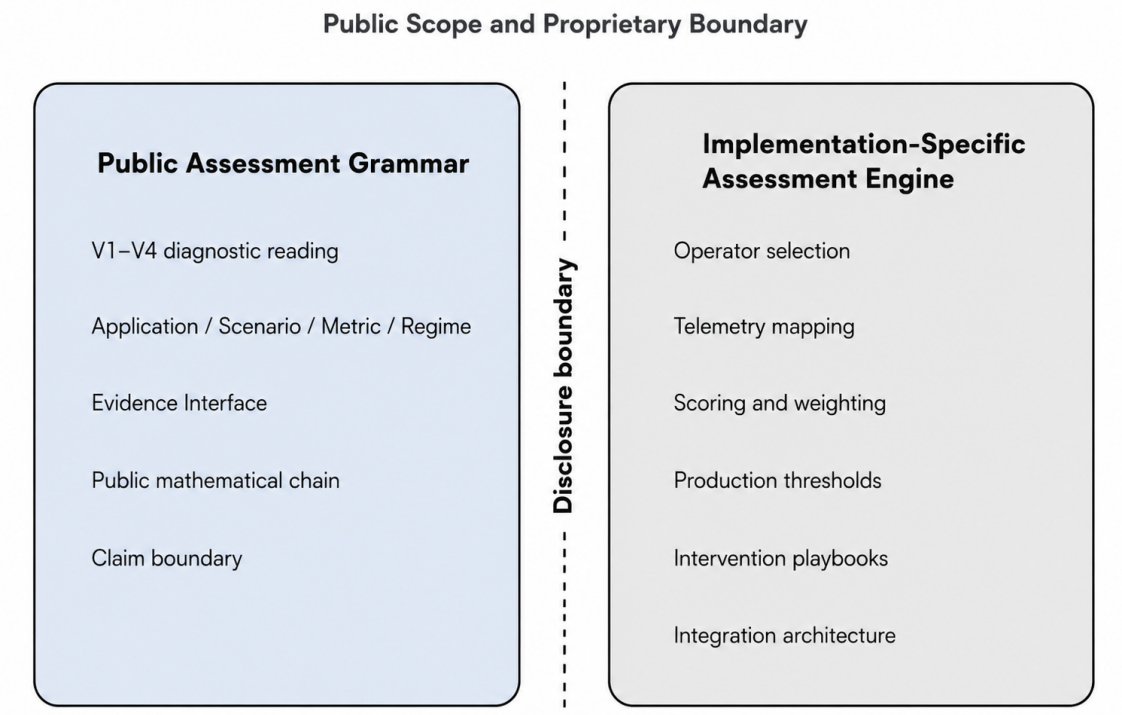

Figure 3. Public versus implementation-specific boundary of the SORT-AI assessment protocol. The present Technical Note formalizes the public assessment grammar, including V1–V4 diagnostic reading, Application-to-Scenario structuring, Metric Set declaration, Regime Classification, and Evidence Interface linkage. Implementation-specific operator selection, telemetry mapping, scoring, weighting, thresholds, intervention playbooks, and production integration remain outside the public scope of this note.

 doi:10.20944/preprints202606.0148.v1

This boundary is essential for the role of the note. The public protocol must be explicit enough to define the structural assessment path, but it must not collapse into a deployment manual, scoring system, or production assessment engine. The result is a public methodological interface: inspectable enough to support scientific discussion while remaining distinct from implementation-specific operational logic.

12. Discussion: Why Structural Assessment Matters for AI Fabrics

AI fabrics are composed execution structures. Their behaviour is distributed across model execution, serving infrastructure, schedulers, orchestrators, runtime control, memory paths, tool chains, policy surfaces, and evidence artefacts. Local metrics, tracing, and observability remain necessary, but they do not by themselves determine whether the composed fabric remains structurally coherent [12,16]. The V_1–V_4 protocol addresses this gap by specifying how an observed condition is structurally read before it is converted into an assessment case.

Treating Applications as assessable regime spaces is the key methodological step. A condition is not collapsed into model quality, infrastructure failure, or an undifferentiated operational incident. Instead, it is assigned to an Application identity, refined into a Scenario Class, associated with a declared Metric Set, and placed within a Core, Boundary, or Overlap regime. This makes cross-layer conditions readable while preserving the distinction between model-centric evaluation and structural assessment [27,30,33].

Boundary and Overlap regimes are particularly important for AI fabrics. Boundary regimes identify conditions near capacity, control, context, SLA, structural, or validity limits. Overlap regimes identify cross-application coupling points where one Application is insufficient to describe the condition alone. In both cases, the assessment protocol renders the condition as an architectural and evidence-relevant state rather than as an isolated anomaly.

The decision relevance of structural assessment extends beyond engineering. Technical states increasingly need to be translated into evidence interfaces and decision surfaces compatible with governance and risk-management frameworks [47–51]. Related work on dangerous-capability and extreme-risk evaluation likewise shows that deployed AI behaviour must be reconstructable, inspectable, and assessable under explicit conditions [52,53]. The present protocol does not prescribe the correct intervention. It makes the assessment input legible.

The relation between the present protocol and neighbouring frameworks is one of complementarity rather than replacement. Each framework continues to address its own concern; SORT-AI provides the public structural reading that connects observed signals to an assessment case and, where applicable, to a downstream Evidence Interface. This relation is summarized in Table 13.

Table 13. Complementarity of the present V1–V4 protocol with neighbouring frameworks. The protocol does not replace any of these layers; it specifies the public structural reading that links an observation to a downstream evidence interface.

Framework / practice	Relation to the present protocol
Observability and tracing	Exposes V1-type signals. The protocol consumes such signals without replacing the observation layer [12].
Site-reliability and incident practice	Organizes operational response. The protocol uses the V1–V4 reading before any operational response is selected [16].
Benchmarking and model evaluation	Evaluates bounded model and evaluation behaviour. The protocol adds a structural reading where deployment behaviour exceeds the benchmark's evaluation scope [30].
Governance and risk-management frameworks	Specify evidence and traceability requirements. The protocol provides a public assessment grammar through which technical states become input-legible to such frameworks [47–49].
Core-3 kernel-damping evidence protocol	Handles declared downstream reproducibility. The protocol passes a compatibility-checked assessment case to that interface [2].

 doi:10.20944/preprints202606.0148.v1

The role of SORT-AI structural assessment is therefore specific and technically useful. It does not replace observability, tracing, benchmarking, governance frameworks, or production monitoring. It defines a public grammar through which their signals can be read as structured assessment cases when behaviour emerges from cross-layer composition.

13. Limitations

The present note has a deliberately narrow scope. It formalizes a public diagnostic protocol and assessment grammar; it does not report a deployed system, production experiment, benchmark comparison, or vendor telemetry study.

First, the protocol is not a production validation of SORT-AI and is not a benchmark or performance comparison. It describes how an AI-fabric observation can be translated into a structural assessment case, but it does not demonstrate deployment performance, operational reliability, or production effectiveness, and it does not claim that SORT-AI outperforms existing observability systems, tracing systems, benchmarks, runtime platforms, or governance frameworks. Its contribution is methodological: the public assessment grammar and its interface to reproducible evidence protocols.

Second, the note does not disclose a runtime implementation, a commercial assessment engine, or a new MOCK version. No implementation-specific operator chain, customer telemetry mapping, weighting function, scoring procedure, production threshold, intervention playbook, or production integration architecture is provided. MOCK v4 remains outside the present methodological contribution and is not treated here as an execution engine. The public versus implementation-specific boundary is stated explicitly in Section 11.

Third, kernel-damping compatibility is referenced, not re-derived. The Core-3 kernel-damping evidence note defines the downstream reproducible evidence protocol [2]; the present note only specifies how a structurally assessed Scenario Class with a declared Metric Set can connect to such an Evidence Interface. Reproducibility statements about the kernel-damping protocol belong to that evidence note and are not duplicated here. The note likewise does not claim structural necessity of SORT for AI-fabric analysis: it defines a public SORT-AI protocol for structural assessment and states the conditions under which a case can be made assessable within that protocol.

14. Conclusion

This Technical Note formalizes the public assessment path through which an observed AI-fabric condition becomes a structured SORT-AI assessment case. Its contribution is neither a new domain architecture nor a second evidence protocol. It is the diagnostic bridge between the canonical SORT-AI Domain Paper and the Core-3 Kernel-Damping Evidence Note: the former supplies the architectural grammar, the latter supplies the reproducible evidence interface, and the present note specifies the public movement between them.

Concretely, the note defines the public assessment path summarized in Eq. 10. The path specifies how a structural condition becomes assessable without reducing an Application to a use case, without collapsing Scenario Classes into new Applications, and without disclosing implementation-specific operator selection, telemetry mapping, weighting logic, scoring functions, thresholds, or intervention playbooks.

The note therefore clarifies why SORT-AI is technically useful beyond the naming of recurrent structural problem forms. It provides a repeatable public path from AI-fabric observation to structured assessment while preserving the distinction between public assessment grammar and implementation-specific assessment execution. Together, the canonical Domain Paper, the present V_1–V_4 Diagnostic Protocol, and the Core-3 Evidence Note define an architecture, a public assessment protocol, and a reproducible evidence interface.

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

 doi:10.20944/preprints202606.0148.v1

Funding: This research received no external funding.

Data Availability Statement: This Technical Note formalizes a public diagnostic protocol and does not depend on empirical data, vendor telemetry, or benchmark measurements. The companion canonical SORT-AI Domain Paper [1] and the Core-3 Kernel-Damping Evidence Note [2] provide the architectural reference and the reproducible evidence interface, respectively. The reproducibility scope of the present note is limited to the public diagnostic grammar and assessment chain described 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, diagnostic formulations, and theoretical claims, was produced and verified by the author, who takes full responsibility for the content of this manuscript.

References

1. Wegener, G. H. (2026). SORT-AI: Domain Architecture and Structural Diagnostics for Advanced AI Systems (Version 4). *MDPI Preprints*. doi:10.20944/preprints202512.1334.v4
2. 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
3. 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
4. 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
5. Wegener, G. H. (2026). SORT-AI: Agentic System Stability in Large-Scale AI Systems — Structural Causes of Cost, Instability, and Non-Determinism in Multi-Agent and Tool-Using Workflows. *MDPI Preprints*. doi:10.20944/preprints202601.1741.v1
6. Dean, J.; Barroso, L. A. (2013). The Tail at Scale. *Communications of the ACM*, **56**(2), 74–80. doi:10.1145/2408776.2408794
7. 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
8. 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`
9. Jouppi, N. P.; Young, C.; Patil, N.; Patterson, D.; 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
10. Verma, A.; Pedrosa, L.; Korupolu, M.; Oppenheimer, D.; Tune, E.; Wilkes, J. (2015). Large-Scale Cluster Management at Google with Borg. *Proceedings of the 10th European Conference on Computer Systems (EuroSys)*. doi:10.1145/2741948.2741964
11. 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
12. Sigelman, B. H.; Barroso, L. A.; Burrows, M.; Stephenson, P.; et al. (2010). Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Google Technical Report. `research.google/pub/pub36356`
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 2019 USENIX Annual Technical Conference*, 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-best-practices`
16. 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.

 doi:10.20944/preprints202606.0148.v1

17. Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; et al. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*. doi:10.1145/3600006.3613165
18. Patel, P.; Choukse, E.; Zhang, C.; Goiri, Í.; et al. (2024). Splitwise: Efficient Generative LLM Inference Using Phase Splitting. *Proceedings of the 51st International Symposium on Computer Architecture (ISCA)*. doi:10.1109/ISCA59077.2024.00019
19. Zhong, Y.; Liu, S.; Chen, J.; Hu, 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`
20. 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`
21. 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)*.
22. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. (2018). Deep Reinforcement Learning That Matters. *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*. doi:10.1609/aaai.v32i1.11694
23. Hutchinson, B.; Rostamzadeh, N.; Greer, C.; Heller, K.; Prabhakaran, V. (2022). Evaluation Gaps in Machine Learning Practice. *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*. doi:10.1145/3531146.3533233
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems 30 (NeurIPS)*.
25. Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; et al. (2020). Language Models Are Few-Shot Learners. *Advances in Neural Information Processing Systems 33 (NeurIPS)*.
26. Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; et al. (2024). The Llama 3 Herd of Models. *arXiv preprint*. arXiv:2407.21783
27. Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; et al. (2021). On the Opportunities and Risks of Foundation Models. *arXiv preprint*. arXiv:2108.07258
28. Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; et al. (2022). Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research (TMLR)*. arXiv:2206.07682
29. 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
30. 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
31. Liang, P.; Bommasani, R.; Lee, T.; Tsipras, D.; et al. (2023). Holistic Evaluation of Language Models. *Transactions on Machine Learning Research (TMLR)*. arXiv:2211.09110
32. Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Raji, I. D.; Gebru, T. (2019). Model Cards for Model Reporting. *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 220–229. doi:10.1145/3287560.3287596
33. Maslej, N.; Fattorini, L.; Perrault, R.; et al. (2025). *The AI Index 2025 Annual Report*. Stanford Institute for Human-Centered Artificial Intelligence.
34. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. *International Conference on Learning Representations (ICLR)*. `openreview.net/forum?id=WE_vluYUL-X`
35. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; et al. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. *Advances in Neural Information Processing Systems 36 (NeurIPS)*. arXiv:2302.04761
36. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. (2023). Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint*. arXiv:2305.16291
37. Park, J. S.; O'Brien, J. C.; Cai, C. J.; Morris, M. R.; Liang, P.; Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*. doi:10.1145/3586183.3606763
38. Wilkinson, M. D.; Dumontier, M.; Aalbersberg, I. J.; Appleton, G.; et al. (2016). The FAIR Guiding Principles for Scientific Data Management and Stewardship. *Scientific Data*, **3**(1), 160018. doi:10.1038/sdata.2016.18

 doi:10.20944/preprints202606.0148.v1

39. Stodden, V.; McNutt, M.; Bailey, D. H.; Deelman, E.; Gil, Y.; Hanson, B.; Heroux, M. A.; Ioannidis, J. P. A.; Taufer, M. (2016). Enhancing Reproducibility for Computational Methods. *Science*, **354**(6317), 1240–1241. doi:10.1126/science.aah6168
40. Sandve, G. K.; Nekrutenko, A.; Taylor, J.; Hovig, E. (2013). Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, **9**(10), e1003285. doi:10.1371/journal.pcbi.1003285
41. Smith, A. M.; Katz, D. S.; Niemeyer, K. E.; FORCE11 Software Citation Working Group (2016). Software Citation Principles. *PeerJ Computer Science*, 2, e86. doi:10.7717/peerj-cs.86
42. Peng, R. D. (2011). Reproducible Research in Computational Science. *Science*, **334**(6060), 1226–1227. doi:10.1126/science.1213847
43. Donoho, D. L. (2010). An Invitation to Reproducible Computational Research. *Biostatistics*, **11**(3), 385–388. doi:10.1093/biostatistics/kxq028
44. National Academies of Sciences, Engineering, and Medicine (2019). *Reproducibility and Replicability in Science*. The National Academies Press, Washington, DC. doi:10.17226/25303
45. Association for Computing Machinery (2020). *Artifact Review and Badging* (Version 1.1). ACM Policy Document. `acm.org/publications/policies/artifact-review-and-badging-current`
46. Katz, D. S.; Gruenpeter, M.; Honeyman, T. (2021). Taking a Fresh Look at FAIR for Research Software. *Patterns*, **2**(3), 100222. doi:10.1016/j.patter.2021.100222
47. 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
48. 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
49. 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 2024/1689. eur-lex.europa.eu/eli/reg/2024/1689
50. Bengio, Y.; Clare, S.; Prunkl, C.; Murray, M.; et al. (2026). *International AI Safety Report 2026*. Department for Science, Innovation and Technology (DSIT).
51. Anderljung, M.; Barnhart, J.; Korinek, A.; Leung, J.; O'Keefe, C.; Whittlestone, J.; et al. (2023). Frontier AI Regulation: Managing Emerging Risks to Public Safety. *arXiv preprint*. arXiv:2307.03718
52. Shevlane, T.; Farquhar, S.; Garfinkel, B.; Phuong, M.; et al. (2023). Model Evaluation for Extreme Risks. *arXiv preprint*. arXiv:2305.15324
53. Phuong, M.; Aitchison, M.; Catt, E.; Cogan, S.; et al. (2024). Evaluating Frontier Models for Dangerous Capabilities. *arXiv preprint*. arXiv:2403.13793
54. Kato, T. (1995). *Perturbation Theory for Linear Operators* (Reprint of 1980 ed.). Springer-Verlag. ISBN 978-3-540-58661-6.
55. Bhatia, R. (1997). *Matrix Analysis*. Springer. ISBN 978-0-387-94846-1.