Article

Advanced AI Systems as Structural Diagnostic Domains: Why Model-Centric Evaluation No Longer Captures System Behavior at Scale

Gregor Herbert Wegener

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

Abstract

Advanced AI systems are increasingly evaluated through model-centric and component-local indicators such as benchmark performance, inference throughput, accelerator utilization, latency, error rates, and local observability signals. These indicators remain necessary, but they become insufficient when the operationally relevant behavior of the system emerges from structural composition rather than from any single component. This paper positions advanced AI systems as structural diagnostic domains and introduces SORT-AI as the canonical AI-domain realization of the broader SORT framework. SORT is used here as a Level-0 structural diagnostic framework: it does not introduce new AI algorithms, runtime mechanisms, physical laws, degrees of freedom, or empirical parameters. Instead, it provides a reading architecture for composed systems whose behavior depends on coupling, control interaction, temporal adaptation, emergent coordination, and evidence requirements. The paper develops four structural paradoxes of advanced AI: healthy local metrics with escalating system cost, locally correct control with globally incoherent behavior, successful retries with growing effective load, and benchmark success with deployment drift. It then explains the four-axis SORT-AI architecture of Domain, Cluster, Application, and Structural Dimensions V1 to V4. A central distinction is that SORT-AI Applications are not use cases, but recurrent structural problem forms within the domain. Runtime Control Coherence is used as a compressed diagnostic reading, and the Sovereign projection is introduced as the meta-domain layer through which technical structural findings become relevant for strategy, governance, auditability, and institutional decision-making. The resulting perspective reframes advanced AI systems not only by model capability, but by the structural coherence of the systems in which those models operate.

Keywords: SORT-AI; domain architecture; structural diagnostics; advanced AI systems; composed systems; runtime control coherence; agentic system stability; interconnect stability; evaluation–deployment divergence; auditability; sovereign AI; hyperscale AI infrastructure

---

1. Executive Framing

Advanced AI systems are still frequently evaluated through model-centric and component-local lenses. Benchmark scores, tokens per second, accelerator utilization, latency, error rates, and local observability signals remain necessary indicators for engineering practice. They support component-level diagnosis, performance regression detection, infrastructure monitoring, and hardware-fault analysis. However, these indicators become incomplete once the operationally relevant behavior of the system emerges from structural composition rather than from any single component [2,10,11].

The relevant object of analysis is therefore no longer the isolated model alone. A modern advanced AI system is a composed structure spanning model execution, serving layers, runtimes, schedulers, orchestrators, control planes, policy-enforcement mechanisms, tool chains, memory paths, deployment

boundaries, and evidence surfaces [2]. Each of these layers may remain locally functional while the composed system develops behavior that is not visible in any one layer. Cost escalation, retry amplification, latency instability, benchmark-to-deployment divergence, and reduced auditability can therefore arise without requiring a discrete component failure.

This shift changes the diagnostic problem. The decisive question is not only whether each subsystem functions within expected parameters, but whether the composed system remains structurally coherent under load, adaptation, control interaction, and deployment pressure. In this regime, local correctness does not guarantee global coherence. A scheduler may optimize according to its own objective, a runtime may satisfy local performance constraints, a policy layer may enforce a valid rule set, and a model may preserve benchmark performance, while the integrated system still exhibits structurally incoherent behavior.

This paper introduces SORT-AI as a structural diagnostic domain for reading advanced AI systems under these conditions. It is positioned as a domain-level technical note, not as an application paper, a classical use case, or an implementation guide. Its purpose is to translate the SORT-AI domain architecture into an engagement-level argument for AI infrastructure, platform, governance, and strategic decision audiences. The intended readers include hyperscaler engineering leads, AI infrastructure architects, AI platform teams, frontier AI laboratories, enterprise AI governance functions, and strategic AI infrastructure decision-makers.

The contribution of this paper is threefold. First, it translates the SORT-AI domain architecture into an engagement-level diagnostic argument for AI infrastructure, platform, governance, and strategic decision audiences. Second, it clarifies why SORT-AI Applications are recurrent structural problem forms rather than use cases. Third, it shows how technical structural diagnosis can become decision-relevant beyond engineering through the Sovereign projection. The paper consolidates the engagement-level reading of a developing SORT-AI research line and connects the canonical domain architecture to decision-facing infrastructure and governance questions.

The analytical object of advanced AI is no longer the isolated model, but the composed system in which model behavior, runtime control, infrastructure coupling, and evidence surfaces interact.

2. What SORT Is: A Structural Diagnostic Framework

SORT, the Supra-Omega Resonance Theory, is used in this paper as a structural diagnostic framework for analyzing composed systems whose relevant behavior cannot be reduced to isolated components [1]. Its role is not to provide another model-specific metric, benchmark, runtime mechanism, or implementation proposal. Its role is to define a reading layer through which structurally coupled systems can be examined in terms of coherence, projection, boundary behavior, and decision relevance.

In this paper, structural coherence denotes the condition in which locally valid components, control surfaces, adaptation processes, and evidence surfaces remain mutually consistent when composed into system-level behavior.

The framework operates at what the broader SORT research line calls Level-0: a structural description layer that precedes domain-specific dynamics, implementation mechanisms, and empirical optimization procedures. SORT does not introduce new physical laws, new degrees of freedom, or new empirical parameters. It does not replace the dynamics of established models, the telemetry of production systems, or the operational expertise embedded in engineering practice. Its objects are structural rather than dynamical: relations between components, projection surfaces, coherence conditions, coupling regimes, and the stability of composed configurations under transformation.

This distinction is essential for this technical note. In an AI-infrastructure context, SORT should not be interpreted as an observability platform, a scheduler, a benchmark suite, a runtime stack, a governance tool, or a replacement for existing diagnostic practice. Standard diagnostics ask whether components operate within expected local parameters. SORT asks a different question: whether

the composed system remains structurally coherent when locally correct components interact across runtime, control, deployment, adaptation, and evidence surfaces.

Because SORT-AI is not bound to one telemetry stack, benchmark, runtime layer, or governance framework, it can relate their outputs within a shared structural architecture. Its practical role is to make signals from existing diagnostics comparable across coupling, learning, control, emergence, and evidence regimes.

SORT-AI is the AI-domain specialization of this structural diagnostic perspective. The underlying SORT core remains shared across domains, but the interpretation of structural states changes with the system class being analyzed. In the AI domain, the relevant structural states are read through model execution, serving paths, runtime orchestration, scheduler behavior, policy enforcement, agentic workflows, evaluation boundaries, and evidence surfaces [2]. The diagnostic task is therefore not to infer model capability alone, but to determine how composed AI systems preserve or lose structural coherence under scale, heterogeneity, control interaction, adaptation, and institutional accountability requirements.

The positive role of SORT-AI is complementary. It provides a reading architecture for situations in which component-level metrics remain necessary but insufficient. A benchmark may characterize model performance, a monitoring stack may report local system health, and a governance framework may define documentation or risk-management requirements. SORT-AI operates across these layers as a structural interpretation framework, asking how local observations compose into system-level behavior and how that behavior becomes decision-relevant for engineering, governance, and strategic infrastructure planning [2].

SORT-AI is adjacent to, but distinct from, SRE, distributed tracing, control theory, and AI risk-management frameworks. SRE and tracing observe operational signals [18,19]; control theory models control dynamics; governance frameworks define accountability requirements. SORT-AI contributes a domain architecture that relates such signals across structural regimes and decision surfaces.

In this role, SORT remains a structural diagnostic framework: it does not introduce new physical laws, new degrees of freedom, or empirical parameters. Its role in the AI domain is to organize observations from existing engineering diagnostics, observability, benchmarking, and governance frameworks into a shared reading architecture focused on the coherence of the composed system.

3. Why AI Is the Canonical Domain Case for SORT

AI is a canonical domain case for SORT because advanced AI systems concentrate multiple structural regimes within a single operational object. A large-scale AI system may simultaneously exhibit physical and logical coupling across infrastructure layers, temporal adaptation through training and deployment drift, runtime control interaction across schedulers and orchestrators, emergent behavior in agentic or tool-mediated workflows, and evidence requirements imposed by auditability, governance, and regulatory traceability [2]. This concentration makes AI structurally distinctive. The relevant diagnostic object is neither a model alone nor an infrastructure stack alone, but the composed system in which these regimes interact.

For this reason, SORT-AI is not defined as a single application of SORT. It is defined as a structured diagnostic domain. The SORT-AI Domain Architecture Paper organizes this domain along four axes: Domain, Cluster, Application, and Structural Dimensions V1–V4 [2]. Domain fixes the problem space, Cluster identifies the structural regime, Application names a recurrent structural problem form, and V1–V4 provide the diagnostic grammar through which observed phenomena are connected to structural causes, effect spaces, and decision surfaces. This architecture is what allows AI systems to be read as structurally coupled systems rather than as a flat catalog of incidents, benchmarks, or operational scenarios.

The current public SORT-AI catalog contains 52 applications distributed across five clusters. This distribution should be read as a snapshot of public diagnostic emphasis, not as a claim of architectural completeness. The strong concentration in Coupling, Control, and Emergence reflects the dominant

pressure surfaces of present-day AI infrastructure: distributed execution fabrics, runtime orchestration, heterogeneous deployment paths, multi-layer control systems, and agentic workflows [3–5]. These are precisely the regimes in which local metrics often remain meaningful but insufficient, because the operationally relevant behavior is produced by interaction across layers.

Cluster E, Evidence, has a smaller public footprint in the current catalog, but this should not be interpreted as limited strategic relevance. Evidence is the cluster through which technical structural findings become institutionally decision-relevant. It connects structural diagnosis to auditability, regulatory traceability, procurement logic, sovereign deployment questions, and governance defensibility. This is why Evidence becomes particularly important in the Sovereign projection developed later in this paper. In the engineering domain, Evidence may appear as one cluster among others; in strategic and institutional contexts, it often becomes the layer through which technical system behavior can be reconstructed, justified, and acted upon.

Table 1. Current public SORT-AI cluster distribution. Counts are a snapshot of the public application catalog documented in the SORT-AI Domain Architecture Paper [2]. They reflect current public application exposure, not architectural completeness or the strategic weight of each cluster.

Cluster	Structural Focus	Count	Engagement Reading
A — Coupling	Cross-layer dependencies	26	Dependencies between components, paths, and infrastructures shape system behavior.
B — Learning	Temporal adaptation	6	System state changes across training, fine-tuning, evaluation, and deployment.
C — Control	Runtime coordination	9	Locally correct control can become globally incoherent under composition.
D — Emergence	Interaction-driven behavior	10	System behavior emerges from agentic, recursive, or non-linear interaction.
E — Evidence	Auditability and traceability	1	Technical states must become reconstructable, defensible, and decision-relevant.
Total		**52**	

AI is canonical for SORT-AI because it exposes coupling, learning, control, emergence, and evidence as simultaneous structural regimes rather than as separate technical topics.

4. The Four Structural Paradoxes of Advanced AI

The shift from component-local analysis to structural diagnosis becomes visible through a set of recurring paradoxes in advanced AI deployments. These paradoxes are not presented here as failures, defects, or evidence of poor engineering. They are structurally meaningful signatures of composed systems in which local indicators remain valid while the aggregate system behavior becomes difficult to explain through those indicators alone. Each paradox identifies a condition in which the diagnostic object has shifted from the component to the interaction surface between components [2].

The four paradoxes discussed below are operationally familiar to infrastructure, platform, and governance teams: healthy local metrics with rising system cost, locally correct control with globally incoherent behavior, successful retries with growing effective load, and benchmark success with deployment drift. They correspond to different dominant clusters in the SORT-AI domain architecture. Their purpose in this technical note is not to quantify these effects, but to show why advanced AI systems require a structural diagnostic grammar in addition to conventional component-level metrics.

4.1. Good Local Metrics, Escalating System Cost

A first paradox occurs when component-local indicators remain within nominal ranges while the system-level cost per useful output continues to rise. Accelerator utilization, throughput, request

latency, and completion rates may suggest that the system is operating acceptably, yet the realized economic efficiency of the composed system degrades. This pattern is especially relevant in large-scale AI infrastructure and modern LLM serving stacks, where utilization alone does not determine effective capacity and where coordination overhead can absorb a growing share of available resources [13,14,16,17].

The structural issue is not that a single component fails. Rather, coupling and control redistribute pressure across the system in ways that local metrics cannot resolve. Interconnect dependencies, memory-path asymmetry, placement decisions, batching behavior, retry mechanisms, and scheduler interactions can jointly increase the gap between nominal capacity and effective capacity [6]. The system is judged locally, while the economic burden is generated structurally.

In SORT-AI terms, this paradox primarily belongs to Cluster A, Coupling, and Cluster C, Control. It indicates that cost behavior is no longer a simple function of component utilization. It becomes a property of the composed execution fabric.

4.2. Locally Correct Control, Globally Incoherent Behavior

A second paradox arises when multiple control layers behave correctly within their own local objectives while their interaction produces globally incoherent runtime behavior. Schedulers, orchestrators, runtime engines, safety gates, policy layers, and autoscaling mechanisms may each optimize valid local targets under limited visibility. Their composition can nevertheless generate oscillation, retry interference, tail-latency instability, capacity inaccessibility, or non-deterministic execution paths [4,7].

The important point is that global incoherence does not require local malfunction. It can arise from the structural composition of independently correct control surfaces. Each layer may observe only a partial state of the system, operate on different time constants, and apply objectives that are locally rational but globally inconsistent. At scale, these differences can become first-order determinants of system behavior.

In SORT-AI terms, this paradox primarily belongs to Cluster C, Control. Its diagnostic significance is that correctness at the level of individual control loops does not imply coherence at the level of the composed system.

4.3. Successful Retries, Growing Effective Load

A third paradox appears when retry logic improves visible success rates while increasing effective system load. Retries are often locally rational: they can mask transient faults, increase completion probability, and stabilize user-facing success metrics. In composed AI systems, however, layered retries across runtime, tool invocation, scheduler, orchestration, and policy surfaces can interact without a shared coherence constraint. The result can be hidden execution volume, amplified latency, inflated cost, and distorted capacity signals [5,15].

This paradox is especially relevant for agentic and tool-mediated workflows. In such systems, retries may not merely repeat an identical operation. They may trigger additional planning steps, tool calls, context updates, validation loops, or delegated subtasks [21]. What appears locally as error handling can become structurally equivalent to load multiplication. The system continues to produce successful outputs, while the execution burden grows disproportionately.

In SORT-AI terms, this paradox primarily belongs to Cluster C, Control, with propagation into Cluster D, Emergence, when retry interaction couples to planning, delegation, or multi-step tool use. The diagnostic object is not the retry mechanism in isolation, but the composed retry geometry of the system.

4.4. Benchmark Success, Deployment Drift

A fourth paradox occurs when benchmark performance remains strong while deployment behavior drifts. Benchmarks measure model behavior under bounded, repeatable, and often simplified conditions. Production deployment exposes the same model to runtime scheduling, memory pressure, serving constraints, tool use, policy enforcement, user interaction patterns, context persistence, and

evidence requirements. The measured object in evaluation and the operational object in deployment are therefore not structurally identical [8,22,23].

This does not make benchmarks irrelevant. It means that benchmark stability is not equivalent to deployment coherence. A model may perform well under evaluation conditions while the deployed system exhibits cost drift, behavioral variance, evidence gaps, or loss of reproducibility [24,25]. Public AI evaluation literature increasingly recognizes the limitations of benchmark-centric assessment when systems are embedded in complex operational environments [20,22].

In SORT-AI terms, this paradox primarily belongs to Cluster B, Learning, with propagation into Cluster E, Evidence, once the divergence must be reconstructed, justified, or made audit-relevant. The diagnostic problem is not only whether a model performs well, but whether the evaluation context projects coherently into the deployment context.

The contribution of SORT-AI is not the claim that these operational phenomena are unknown. Retry amplification, tail-latency effects, control-loop interference, and evaluation–deployment gaps are already familiar across systems engineering, distributed systems, and AI evaluation literature. The contribution is to read these otherwise separate phenomena through a shared domain architecture, so that they become comparable as recurrent structural forms across coupling, learning, control, emergence, and evidence regimes.

These paradoxes do not indicate that advanced AI systems are failing. They indicate that the dominant diagnostic object has shifted from component behavior to structural composition.

5. The Four-Axis Architecture

SORT-AI organizes the AI domain along four interacting axes: Domain, Cluster, Application, and Structural Dimensions V1 to V4 [2]. These axes define the ordering logic through which advanced AI systems become diagnostically readable as composed systems. Domain fixes the problem space, Cluster identifies the structural regime, Application names the recurrent structural problem form, and V1 to V4 define the diagnostic sequence through which observations are connected to structural causes, effect spaces, and decision relevance.

No single axis is sufficient on its own. Domain alone would remain too broad, because "advanced AI systems" includes infrastructure, models, runtimes, agentic workflows, deployment environments, and governance surfaces. Application alone would produce a growing list of named problem forms without a stable architecture. Diagnostic dimensions alone would describe a method of reading but would not determine the system class or structural regime being read. The four axes therefore operate together as the backbone of SORT-AI.

This section presents the architecture at engagement level only. It does not reproduce the operator algebra, kernel definitions, projection-space formalism, or mathematical basis of the underlying SORT framework. Those elements remain part of the scientific domain paper and the broader SORT framework documentation [1,2]. The purpose here is narrower: to explain how the domain architecture makes advanced AI systems readable for infrastructure, platform, governance, and strategic decision audiences.

Figure 1. The four-axis SORT-AI architecture, read from problem space to decision surface. Domain fixes what is analyzed; Cluster fixes the structural regime; Application names the recurrent structural form; V1–V4 provide the diagnostic grammar; the decision surface follows from the diagnostic reading.

5.1. Domain: What Is Being Analyzed

The Domain axis identifies the problem space to which SORT is applied. In the present case, that space is the class of advanced AI systems understood as composed structures. This includes model execution, serving layers, runtime environments, schedulers, orchestrators, control planes,

policy-enforcement mechanisms, agentic workflows, memory paths, deployment boundaries, and evidence surfaces [2].

The methodological role of the Domain axis is to fix the interpretive frame. The underlying SORT logic remains structurally consistent across domains, but structural states are interpreted differently depending on the system class. In AI, a structural state may correspond to runtime coordination, benchmark-to-deployment divergence, agentic workflow instability, interconnect dependency, or evidence reconstruction. The Domain axis determines that these states are read as AI-system phenomena rather than as generic complex-system or infrastructure events.

This distinction prevents analytical flattening. Without the Domain axis, structurally different systems could be described with the same vocabulary but without domain-specific meaning. By fixing the AI domain explicitly, SORT-AI establishes the context in which coupling, learning, control, emergence, and evidence become operationally interpretable.

5.2. Cluster: How the Problem Takes Structural Form

The Cluster axis identifies the dominant structural regime in which a phenomenon takes form. SORT-AI currently organizes the AI domain into five clusters: Coupling, Learning, Control, Emergence, and Evidence [2]. These clusters are not topic folders. They separate structurally distinct regimes of system behavior.

Coupling describes conditions in which dependencies between components, paths, layers, or infrastructures shape system behavior. Learning captures temporal adaptation, drift, and evaluation-to-deployment divergence. Control describes the interaction of schedulers, orchestrators, runtimes, policy layers, and other operative decision surfaces. Emergence captures behavior that arises from recursive, agentic, tool-mediated, or non-linear interaction. Evidence addresses the conditions under which technical states become traceable, reconstructable, auditable, and decision-relevant.

Real systems may exhibit cross-cluster propagation. A benchmark-to-deployment divergence may begin as a Learning problem, propagate into Control through runtime adaptation, and become an Evidence problem once the divergence must be reconstructed for audit or governance purposes. The Cluster axis therefore fixes the dominant entry regime of the diagnosis; it does not imply that the phenomenon is isolated from all other clusters.

5.3. Application: Which Recurrent Structural Form Appears

The Application axis identifies the recurrent structural problem form within a cluster. This point is central to the SORT-AI architecture. An Application is not a business use case, deployment scenario, customer story, or isolated operational incident. It is a named structural form that can recur across different systems, infrastructures, and institutional contexts [2].

For example, Runtime Control Coherence is not merely a runtime example. It is a recurrent Control-class problem form in which locally correct control surfaces can compose into globally incoherent system behavior. Interconnect Stability is not merely an infrastructure example. It is a recurrent Coupling-class problem form in which physical, topological, and synchronization dependencies reshape effective system capacity. Agentic System Stability is not merely an agent example. It is a recurrent Emergence-class problem form in which tool use, planning loops, delegation, and multi-step interaction generate system-level instability.

This architecture makes the domain extensible. New Applications may be added when a recurrent structural condition becomes identifiable, but they do not redefine the domain itself. The catalog grows by additive extension, while the organizing grammar remains stable. This is what distinguishes a domain architecture from a collection of use cases.

5.4. V1 to V4: How Diagnosis Moves from Observation to Decision

Structural Dimensions V1 to V4 define the diagnostic grammar of SORT-AI. They explain how an observed phenomenon becomes structurally interpretable and decision-relevant. V1 denotes the observed structural phenomenon. V2 identifies the structural cause, coupling, or interaction regime.

V3 describes the structural effect space in which the phenomenon becomes intelligible. V4 identifies the decision and utilization space that follows from the diagnostic reading [2].

The V1 to V4 sequence is important because it prevents diagnosis from stopping at symptom description. A latency anomaly, retry pattern, cost increase, benchmark divergence, or auditability gap is not treated merely as an observed event. It is read through a structured sequence: what is visible, what structural relation produces it, what effect space it belongs to, and what decision surface it informs.

The grammar is invariant in form and domain-specific in content. The same V1 to V4 sequence can be applied to an interconnect problem, a runtime-control problem, an agentic-workflow problem, or an evidence-reconstruction problem. What changes is the interpretation of the structural state, not the diagnostic sequence itself. This gives SORT-AI repeatability without reducing different phenomena to the same explanation.

At this architectural level, SORT-AI remains a diagnostic reading framework: it references the underlying mathematical basis without reproducing it and focuses on how composed AI systems become structurally readable for engineering and decision contexts. The four-axis architecture makes AI systems readable without reducing them either to isolated model behavior or to a flat list of operational incidents.

6. Applications Are Not Use Cases

A central distinction in SORT-AI is the difference between an Application and a use case. This distinction is not terminological. It determines the level at which the domain is organized. A use case usually describes a context-specific deployment, customer scenario, operational need, or business-facing application of a capability. A SORT-AI Application has a different function. It denotes a recurrent structural problem form within the AI domain [2].

This is why the present manuscript is positioned as a domain-level technical note rather than as a classical use case paper. Its purpose is not to describe one operational scenario in which SORT-AI may be applied. Its purpose is to explain why advanced AI systems constitute a diagnostic domain in which multiple recurrent structural problem forms can be identified, compared, and extended. The domain is therefore not built from use cases upward. It is organized through a structural grammar that allows specific applications to be derived and placed.

Use cases describe where a problem appears. SORT-AI Applications describe the structural form by which a problem recurs across deployment contexts. The same structural problem form may appear in different infrastructures, organizations, architectures, or operational regimes. Conversely, two superficially similar deployment scenarios may belong to different SORT-AI Applications if their dominant structural causes, effect spaces, or decision surfaces differ. This distinction prevents the domain from becoming a loose catalog of examples.

A phenomenon qualifies as a SORT-AI Application when it satisfies five conditions. It recurs across more than one deployment context; it can be assigned to a dominant structural cluster; it remains readable through the V1–V4 diagnostic grammar; it is not dependent on a single vendor, product, or implementation; and it opens a recurrent decision surface. If these conditions are not met, the phenomenon is better treated as a use case, scenario, incident, or implementation-specific observation rather than as a domain Application.

The distinction also protects the stability of the domain architecture. If each new deployment pattern were treated as a separate use case, the domain would fragment as the AI ecosystem evolves. New serving architectures, orchestration frameworks, agent designs, governance requirements, or deployment environments would continually generate new scenario labels. SORT-AI avoids this by organizing the domain around recurrent structural forms. New Applications may be added when a stable problem form becomes identifiable, but established Applications are not redefined merely because the operational setting changes.

The distinction also supports comparability. Different systems can be compared through their underlying structural form rather than only through their surface symptoms. A cost increase in a heterogeneous inference fabric, an instability in a multi-agent workflow, and a drift pattern in benchmark-to-deployment transfer are not comparable as business scenarios. They become comparable only when each is read through the same architectural grammar: Domain, Cluster, Application, and V1 to V4.

The Core-3 Applications illustrate this logic. AI.01 is not simply an infrastructure example. It is the Coupling-class form of Interconnect Stability, where physical, topological, and synchronization dependencies reshape the effective behavior of distributed AI infrastructure [3]. AI.04 is not simply a runtime example. It is the Control-class form of Runtime Control Coherence, where locally correct schedulers, orchestrators, runtime engines, and policy layers can compose into globally incoherent behavior [4]. AI.13 is not simply an agent example. It is the Emergence-class form of Agentic System Stability, where planning, tool invocation, delegation, and multi-step interaction generate system-level behavior that cannot be reduced to any single agent step [5].

This architecture makes the SORT-AI catalog extensible without making it unstable. New Applications can supplement the existing domain when additional recurrent structural problem forms become visible. However, they must remain consistent with the cluster architecture and the V1–V4 diagnostic grammar. The result is a domain that can expand with the evolution of advanced AI systems while preserving a stable analytical structure.

Use cases describe where a problem appears. SORT-AI Applications describe the structural form by which the problem recurs.

7. The Core-3 as Entry Points, Not the Whole Domain

Within the current SORT-AI architecture, AI.01, AI.04, and AI.13 function as Core-3 entry points. They are useful because each isolates a structurally distinct and operationally visible pressure regime: Coupling, Control, and Emergence. Together, they provide a compact way to enter the domain without requiring the reader to traverse the full application catalog at the outset [2].

The Core-3 should not be mistaken for the domain itself. They are orientation points, not a boundary definition. AI.01 introduces the Coupling regime through Interconnect Stability. AI.04 introduces the Control regime through Runtime Control Coherence. AI.13 introduces the Emergence regime through Agentic System Stability. Each application is compatible with the same V1–V4 diagnostic grammar, but each enters the domain through a different dominant structural class.

Table 2. The Core-3 as entry points into the SORT-AI domain. Each application instantiates a distinct dominant cluster while remaining readable through the same V1–V4 diagnostic grammar [2].

Application	Cluster	Role
AI.01	A — Coupling	Interconnect Stability as physical and topological coupling in distributed AI infrastructure [3].
AI.04	C — Control	Runtime Control Coherence as logical control coupling across scheduler, orchestrator, runtime, and policy layers [4].
AI.13	D — Emergence	Agentic System Stability as semantic and agentic coupling across multi-step, tool-mediated workflows [5].

The Core-3 are deliberately selected because they expose three of the most visible stress surfaces in current AI systems. AI.01 captures the shift from isolated compute performance to coupled infrastructure behavior. AI.04 captures the shift from local control correctness to system-level control coherence. AI.13 captures the shift from single-step inference to extended, tool-mediated, and interaction-driven execution. These three entry points are therefore well suited for engagement with AI infrastructure, platform, and agent-system audiences.

However, the domain architecture is broader than the Core-3. Cluster B, Learning, is not reducible to Coupling, Control, or Emergence. It captures temporal adaptation, evaluation–deployment divergence, benchmark saturation, model-update effects, and drift patterns that modify the structural condition of the system over time [8]. Evaluation–deployment projection instability is one example of a Learning-class form: a model behaves consistently under evaluation conditions while its deployed behavior drifts as runtime, tool use, and context conditions evolve. These phenomena may interact with runtime or infrastructure layers, but their dominant entry regime is temporal transformation rather than physical coupling or control interaction.

Cluster E, Evidence, is also not reducible to the Core-3. It addresses the conditions under which technical system states become auditable, reconstructable, defensible, and institutionally decision-relevant. Deployment-drift evidence aggregation is one example of an Evidence-class form: its primary function is not to detect drift in the operational sense, but to make drift reconstructable and decision-relevant for governance, audit, or procurement actors. Evidence may have a smaller public application footprint in the current catalog, but it carries disproportionate strategic importance for regulated AI systems, frontier governance, procurement, compliance, and Sovereign projection. Many technical findings become strategically useful only once they can be translated into evidence surfaces.

The Core-3 therefore provide a practical entry route into SORT-AI, not a complete representation of the domain. They demonstrate how the framework reads Coupling, Control, and Emergence, while Learning and Evidence preserve additional diagnostic regimes that become central in other contexts. This distinction is important for preventing a common misunderstanding: SORT-AI is not the sum of three core papers. It is a domain architecture within which those papers become mutually interpretable.

The Core-3 provide strong entry points into the SORT-AI domain. They do not define its boundary.

8. Compressed Diagnostic Reading: Runtime Control Coherence

Runtime Control Coherence, represented as AI.04 within the SORT-AI domain, is used here as a compressed diagnostic reading because it illustrates the central diagnostic claim of this paper with particular clarity: locally correct control surfaces can compose into globally incoherent system behavior [4]. The example is useful because runtime control is familiar to AI infrastructure and platform teams. Schedulers, orchestrators, runtime engines, policy gates, retry mechanisms, and service-level controls are already recognized as operationally relevant layers. SORT-AI does not replace these layers or prescribe how they should be implemented. It reads their interaction as a structural condition.

This section is not a use case paper for AI.04. The mechanism-level treatment of Runtime Control Coherence is documented separately [4,7]. The purpose here is narrower: to show how one SORT-AI Application can be read through the V1–V4 diagnostic grammar without reducing the application to a single scenario. Runtime Control Coherence remains one structural problem form even when it appears through different operational patterns, including multi-layer control stacks, retry-heavy inference, and SLA-adjacent runtime behavior. The scenario readings developed below are diagnostic constructions rather than empirical case studies; their purpose is to demonstrate transferability of the V1–V4 grammar across different operational appearances of the same Application.

The example also clarifies why Applications are not use cases. AI.04 does not describe one product surface, deployment environment, or customer situation. It identifies a recurrent Control-class condition in which multiple locally valid control decisions interact without sufficient global coherence. The same structural form can appear in inference serving, multi-tenant AI platforms, agentic workflow execution, policy-constrained runtime systems, or heterogeneous infrastructure environments.

8.1. V1: Observed Phenomenon

At the V1 level, Runtime Control Coherence begins with observed system behavior. The relevant phenomena are not necessarily hard failures. They may appear as cost escalation despite healthy local metrics, tail-latency instability under nominally stable utilization, non-deterministic execution at the runtime layer, or fluctuating effective capacity without visible component failure [6,16]. In

many cases, the system continues to serve requests, satisfy local success criteria, and report acceptable component-level health.

This is precisely why the phenomenon is structurally important. A component-local diagnosis may not identify a fault because no individual component is clearly malfunctioning. The scheduler may continue to schedule, the runtime may continue to execute, the orchestrator may continue to coordinate, and the policy layer may continue to enforce valid constraints. Yet the composed system may exhibit instability, rising cost per useful output, retry amplification, or reduced reproducibility.

For AI.04, the V1 observation is therefore not simply that runtime behavior is variable. The observed condition is that variability persists despite apparently valid local operation. The diagnostic object is the discrepancy between local correctness and global behavior.

8.2. V2: Structural Cause

At the V2 level, the observed phenomenon is read as the result of structural interaction between independently correct control surfaces. These surfaces may include schedulers, orchestrators, runtime engines, autoscaling mechanisms, policy layers, retry logic, routing decisions, admission controls, and SLA-adjacent mechanisms [4,7]. Each layer may operate according to a valid local objective, but the objectives are not necessarily aligned at the level of the composed system.

The structural cause is therefore not a single defective control mechanism. It is the interaction geometry of control. Different layers may observe different subsets of system state, operate on different time constants, respond to different thresholds, and optimize different local targets. A scheduler may attempt to improve resource allocation, while an autoscaler modifies capacity, while retry logic increases execution pressure, while a policy layer constrains routing choices. Each decision may be rational locally, yet the combined system may drift into incoherence.

This reading is consistent with a broader systems principle: local correctness does not automatically aggregate into global coherence when functions are distributed across layers with partial visibility and divergent control assumptions [12]. In SORT-AI terms, V2 identifies the structural cause as control-layer composition under limited shared coherence.

8.3. V3: Structural Effect Space

At the V3 level, SORT-AI identifies the structural effect space in which the observed condition becomes intelligible. For Runtime Control Coherence, this effect space includes control-loop interference, retry amplification, tail-latency instability, capacity inaccessibility, and reduced reproducibility [4,6]. These are not separate incidents. They are related expressions of a common structural condition: the composed runtime is governed by interacting control decisions whose aggregate behavior is not globally coherent.

Control-loop interference occurs when multiple layers respond to related signals without coordination across their response functions. Retry amplification occurs when local error-handling mechanisms increase effective system load faster than the system can absorb it. Tail-latency instability occurs when control responses shift execution pressure into long-tail behavior rather than resolving the underlying structural condition. Capacity inaccessibility occurs when nominal resources exist but cannot be effectively used because control, placement, policy, or coordination constraints prevent coherent utilization. Reduced reproducibility occurs when repeated executions traverse different runtime paths under similar apparent conditions.

The V3 reading is therefore not merely descriptive. It groups otherwise scattered operational symptoms into a single structural effect space. This is the diagnostic value of the application: it converts a set of visible symptoms into a coherent structural interpretation.

8.4. V4: Decision and Utilization Space

At the V4 level, the structural reading becomes decision-relevant. Runtime Control Coherence informs architecture-level decisions about coordination boundaries between control surfaces, runtime observability priorities, governance interpretation, SLO design, procurement assumptions, and capac-

ity planning [2,6]. The point is not to prescribe a specific implementation, but to clarify what kind of decision space the diagnosis opens.

For architecture teams, the relevant question becomes whether control layers are independently optimized or structurally coordinated. For platform teams, the question becomes whether observability captures only local metrics or also cross-layer control interaction. For governance teams, the question becomes whether system behavior can be reconstructed when outcomes depend on runtime path variation. For procurement and capacity planning, the question becomes whether nominal capacity can be treated as usable capacity under the actual control geometry of the deployment.

In this sense, V4 links the structural diagnosis back to utilization. A system may have sufficient hardware, competent local control mechanisms, and valid monitoring surfaces while still lacking an explicit way to interpret the coherence of its control composition. AI.04 makes that missing decision layer visible.

8.5. Three Scenario Readings

The same Application can appear through different operational scenarios. This is why AI.04 is an Application rather than a use case. The scenarios below are not separate Applications. They are three readings of one recurrent Control-class structural problem form.

S1 — Multi-Layer Control Stack.

In a multi-layer control stack, several control surfaces optimize locally at the same time. A scheduler assigns workloads, an orchestrator manages placement and lifecycle, a runtime engine adapts execution behavior, a policy layer enforces constraints, and an autoscaling mechanism modifies capacity. Each layer can be correct within its own scope. However, if their response functions interact without a shared coherence model, the runtime can enter a regime of circular interference across scheduler, orchestrator, runtime, and policy layers [4,18]. The observable result may be unstable throughput, inconsistent latency, or capacity that appears available but cannot be used coherently.

S2 — Retry-Heavy Inference.

In retry-heavy inference, retry mechanisms stabilize visible success rates while increasing effective structural load. A failed or delayed request may trigger one retry at the application layer, another at the runtime layer, and additional compensatory behavior in routing, batching, or tool execution. Locally, retries appear to improve reliability because more requests eventually complete. Structurally, however, success and effective execution burden diverge. The system may become more expensive, more variable, and less reproducible even as visible completion remains acceptable. In this scenario, retry behavior becomes part of the control geometry rather than a simple error-handling mechanism.

S3 — SLA-Adjacent Runtime.

In an SLA-adjacent runtime regime, the system remains nominally near local targets while shifting cost, tail latency, and capacity reserves into an unstable boundary condition. Control surfaces respond aggressively because execution remains close to decision thresholds. Small changes in load, routing, or policy constraints can trigger disproportionate adjustments across scheduling, runtime adaptation, and retry logic. The system is not necessarily failing, but it operates in a narrow band where local compliance can coexist with global instability. SORT-AI reads this as a structural boundary regime: the system remains functional, but its coherence margin is reduced.

These three scenarios illustrate the same point. Runtime Control Coherence is structurally stable as an Application even when its operational appearances differ. The V1–V4 grammar remains invariant across them: observed runtime instability, structural interaction of control surfaces, a control-coherence effect space, and a decision surface involving architecture, observability, governance, and capacity planning.

Bridge to Section 9.

What AI.04 illustrates for the control regime, the following section generalizes across the structural boundaries on which SORT-AI is strongest.

9. Where SORT-AI Is Strongest

SORT-AI becomes most informative at structural boundaries rather than within isolated components. Many of the consequential problems in advanced AI systems first appear as mismatches between local correctness and global behavior: a model remains performant while deployment behavior drifts, a scheduler remains locally effective while runtime coordination becomes unstable, or an infrastructure layer appears healthy while effective capacity becomes inaccessible [2]. These conditions are difficult to diagnose through component-local metrics because the relevant effect is produced between layers.

This positioning does not reduce the importance of conventional diagnostics. Component-level observability, performance monitoring, fault detection, benchmark evaluation, and governance controls remain necessary. SORT-AI adds a complementary structural layer for cases in which the relevant question is not whether one component is operating correctly, but whether multiple locally correct components remain coherent when composed into a larger system. Its strongest role is therefore at the interfaces where model behavior, runtime execution, infrastructure coupling, control policy, deployment context, and evidence requirements interact.

Table 3. Structural boundaries on which SORT-AI becomes most informative. Each boundary corresponds to a diagnostic question that local component metrics typically cannot resolve [2].

Boundary	Diagnostic Question
Model / Runtime	How does the execution layer reshape observable model behavior?
Scheduler / Orchestrator	When do locally correct control decisions become globally incoherent?
Evaluation / Deployment	Why does benchmark stability fail to transfer into deployment stability?
Infrastructure / Policy	How do technical and regulatory control surfaces interact?
Technical Behavior / Evidence	When does a system state become auditable, reconstructable, and decision-relevant?

These boundaries also explain why SORT-AI connects naturally to strategic and institutional questions. Once a technical condition crosses from runtime behavior into auditability, procurement, regulatory exposure, or sovereign deployment, it is no longer only an engineering concern. It becomes part of a broader decision surface. The following section develops this transition through the Sovereign projection.

SORT-AI becomes strongest where conventional diagnostics lose explanatory power because the relevant effect is produced between layers.

10. Why This Matters Beyond Engineering: The Sovereign Projection

The structural diagnosis of advanced AI systems does not end at the engineering boundary. In hyperscale, regulated, frontier, or sovereign deployment contexts, technical system behavior becomes relevant to procurement, auditability, regulatory defensibility, institutional accountability, and strategic control. This is the role of SORT-Sovereign within the broader architecture. Sovereign projection denotes the translation of selected technical structural findings into institutional decision categories: dependency, controllability, and reconstructability. SORT-Sovereign is introduced as a meta-domain layer, not as a full technical copy of SORT-AI. It projects selected technical structural findings into strategic, regulatory, and state decision spaces [2,9].

Institutional decision spaces operate through three questions: what dependencies exist, what can be controlled, and what can be reconstructed. These correspond to Coupling, Control, and Evidence. Learning and Emergence remain technically important within SORT-AI, but they become institutionally actionable only once translated into dependency exposure, control problem, or evidence

requirement. The restriction of SORT-Sovereign to Clusters A, C, and E is therefore deliberate. A learning drift, for example, becomes a sovereign or governance issue when it affects dependency structure, operational steerability, or the reconstructability of system behavior. An emergent agentic pattern becomes institutionally actionable when it can be read as a control-coherence problem, an infrastructure exposure, or an evidence gap.

This restriction prevents the Sovereign layer from becoming an expanded technical AI domain. It preserves the distinction between technical diagnosis and institutional projection. SORT-AI reads the composed AI system as a technical structural domain. SORT-Sovereign reads selected outputs of that diagnosis through decision categories that matter for hyperscalers, frontier laboratories, enterprise governance functions, regulators, and public-sector AI infrastructure. This separation is important because strategic decision-makers do not require a complete replica of the technical domain. They require a translation layer that identifies which structural conditions affect dependency, control, evidence, and accountability.

10.1. Sovereign Cluster Scope

Within the Sovereign projection, Cluster A, Coupling, concerns structural dependencies. These include infrastructure dependency, cloud and vendor dependency, interconnect geometry, deployment-spanning coupling risk, data-path exposure, and the degree to which system behavior depends on external or cross-domain execution surfaces. In strategic terms, Coupling determines where technical dependency becomes procurement exposure, concentration risk, or sovereign infrastructure concern.

Cluster C, Control, concerns operational steerability. This includes runtime governance, policy enforcement, control coherence, escalation pathways, intervention points, and the ability to maintain predictable system behavior under load, adaptation, and institutional constraints. In strategic terms, Control determines whether an organization can govern the composed system rather than merely operate its components.

Cluster E, Evidence, concerns reconstructability and defensibility. This includes auditability, compliance, traceability, incident reconstruction, regulatory reporting, and the ability to justify decisions about system behavior after deployment [2,27–29]. In strategic terms, Evidence determines whether technical states can be converted into institutional knowledge.

The importance of this projection is reinforced by the direction of AI governance and risk-management practice. Frameworks such as the NIST AI Risk Management Framework, the NIST Generative AI Profile, the EU AI Act, and frontier AI risk discussions increasingly require organizations to connect technical behavior with documentation, traceability, risk classification, and accountable decision processes [26–30]. SORT-Sovereign does not replace these frameworks. It provides a structural interpretation layer for the technical conditions that must eventually be rendered governable, auditable, and strategically meaningful.

This section does not develop a policy argument; it identifies how structural diagnosis becomes decision-relevant in environments where technical behavior must be governed, audited, procured, or defended.

For hyperscalers and frontier AI organizations, the implication is direct. Structural diagnostics are not only an engineering lens. They become a decision layer between system behavior, governance exposure, and strategic control.

11. Closing Insight

Advanced AI systems are still frequently evaluated through model-centric and component-local lenses. This remains necessary for engineering practice, but it is no longer sufficient for explaining the most consequential behavior of large-scale deployed systems. The relevant effects increasingly emerge through composition: between model and runtime, scheduler and orchestrator, evaluation and deployment, infrastructure and policy, technical behavior and evidence surface.

SORT-AI provides a domain architecture for reading these systems structurally [2]. It does not reduce them to isolated models, disconnected operational incidents, or a flat catalog of use cases. It

reads advanced AI systems as composed structures in which recurrent problem forms arise across Coupling, Learning, Control, Emergence, and Evidence. The purpose of this architecture is not to replace existing diagnostics, but to make visible the structural conditions under which local correctness, benchmark performance, or component-level health no longer explain global system behavior.

This distinction is the central contribution of the domain-level technical-note framing. SORT-AI Applications are not use cases. They are recurrent structural problem forms that make different operational scenarios comparable without flattening them into the same incident category. AI.01, AI.04, and AI.13 provide entry points into Coupling, Control, and Emergence, while Learning and Evidence preserve additional diagnostic regimes that become increasingly important as AI systems adapt over time and become subject to institutional accountability.

The Sovereign projection extends this logic beyond engineering. Once technical behavior must be reconstructed, justified, governed, procured, regulated, or defended, structural diagnosis becomes part of a broader decision surface. In that context, system coherence is not only an operational property. It becomes a strategic condition for auditability, governance, and control.

Advanced AI systems are no longer defined only by model capability, but by the structural coherence of the systems in which those models operate.

Acknowledgments: The author acknowledges prior SORT-AI architecture documents and companion analyses that informed the structural framing of this manuscript.

Conflicts of Interest: The author declares no conflicts 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, diagnostic framing, domain architecture, and conceptual positioning, was produced and verified by the author, who takes full responsibility for the content of this manuscript.

Data Availability Statement: No new data were generated in this study. Referenced materials are available through the cited publications and public source links where applicable.

1. 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
2. Wegener, G.H. (2026). SORT-AI: Domain Architecture and Structural Diagnostics for Advanced AI Systems. *MDPI Preprints*. doi:10.20944/preprints202512.1334.v3
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. *MDPI Preprints*. doi:10.20944/preprints202601.0298.v1
5. Wegener, G.H. (2026). SORT-AI: Agentic System Stability in Large-Scale AI Systems. *MDPI Preprints*. doi:10.20944/preprints202601.1741.v1
6. Wegener, G.H. (2026). SORT-AI: Structural Efficiency Recovery in Hyperscale AI Systems. *MDPI Preprints*. doi:10.20944/preprints202602.0015.v1
7. Wegener, G.H. (2026). Runtime Control Layer as a Hidden Performance Variable: Why Model Performance Is Increasingly Determined by Runtime Control Coherence. *Zenodo*. doi:10.5281/zenodo.19784935
8. Wegener, G.H. (2026). Evaluation–Deployment Structural Divergence in Large-Scale AI Systems: A Diagnostic Perspective on Projection Mismatch and Runtime Drift. *Zenodo*. doi:10.5281/zenodo.19784578
9. Wegener, G.H. (2026). Structural Oversight as a Performance Advantage: Why Frontier AI Governance Should Optimize for Governable Capability, Not Blunt Capability Suppression. *Zenodo*. doi:10.5281/zenodo.20132784
10. 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
11. Dean, J.; Barroso, L.A. (2013). The Tail at Scale. *Communications of the ACM*, 56(2), 74–80. doi:10.1145/2408776.2408794
12. 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

13. 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. *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 521–538. https://www.usenix.org/conference/osdi22/presentation/yu
14. Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C.H.; Gonzalez, J.E.; Zhang, H.; Stoica, I. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. *29th ACM Symposium on Operating Systems Principles (SOSP)*. https://arxiv.org/abs/2309.06180
15. Moritz, P.; Nishihara, R.; Wang, S.; Tumanov, A.; Liaw, R.; Liang, E.; Elibol, M.; Yang, Z.; Paul, W.; Jordan, M.I.; Stoica, I. (2018). Ray: A Distributed Framework for Emerging AI Applications. *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 561–577. https://www.usenix.org/conference/osdi18/presentation/moritz
16. Meta Engineering. (2024). Taming Tail Utilization of Ads Inference at Meta Scale. *Meta Engineering Blog*. https://engineering.fb.com/2024/07/10/production-engineering/tail-utilization-ads-inference-meta/
17. Databricks Engineering. (2024). LLM Inference Performance Engineering: Best Practices. *Databricks Engineering Blog*. https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices
18. Beyer, B.; Jones, C.; Petoff, J.; Murphy, N.R. (eds.) (2016). Site Reliability Engineering: How Google Runs Production Systems. *O'Reilly Media*. https://sre.google/sre-book/table-of-contents/
19. OpenTelemetry Authors. (2024). OpenTelemetry Specification. *Cloud Native Computing Foundation*. https://opentelemetry.io/docs/specs/otel/
20. 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).
21. Wooldridge, M. (2009). An Introduction to MultiAgent Systems, 2nd ed. *John Wiley & Sons*. ISBN 978-0-470-51946-2.
22. 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. https://arxiv.org/abs/2502.06559
23. Liang, P.; Bommasani, R.; Lee, T.; Tsipras, D.; Soylu, D.; Yasunaga, M.; Zhang, Y.; Narayanan, D.; Wu, Y.; Kumar, A.; et al. (2023). Holistic Evaluation of Language Models. *Transactions on Machine Learning Research (TMLR)*. https://arxiv.org/abs/2211.09110
24. Bommasani, R.; Hudson, D.A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M.S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. (2022). On the Opportunities and Risks of Foundation Models. *arXiv preprint* arXiv:2108.07258. https://arxiv.org/abs/2108.07258
25. Maslej, N.; Fattorini, L.; Perrault, R.; et al. (2025). The AI Index 2025 Annual Report. *Stanford Institute for Human-Centered AI (HAI)*. https://hai.stanford.edu/ai-index/2025-ai-index-report
26. 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
27. National Institute of Standards and Technology. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0), AI 100-1. *NIST*. doi:10.6028/NIST.AI.100-1
28. National Institute of Standards and Technology. (2024). Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, AI 600-1. *NIST*. doi:10.6028/NIST.AI.600-1
29. 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. https://eur-lex.europa.eu/eli/reg/2024/1689/oj
30. Bengio, Y.; et al. (2025). International AI Safety Report. *Department for Science, Innovation and Technology (DSIT)*, January 2025.