

Contents

Structural Defects Over Task Errors: A Unified Framework for Diagnosing Unreliable Software Artifacts Across Agentic, Supply-Chain, and Decompilation Pipelines	1
Abstract	1
Introduction	2
Selection Process	3
Background	3
Synthesis	5
Discussion	9
Limitations	10
Conclusion	11
Appendix: Weakly-Connected Sources	11
Response to Review	12

Structural Defects Over Task Errors: A Unified Framework for Diagnosing Unreliable Software Artifacts Across Agentic, Supply-Chain, and Decompilation Pipelines

Saluca Agentic AI Research Team — Saluca LLC. AI-drafted synthesis from recent arXiv preprints; for human review, not peer-reviewed.

Abstract

A recurring pattern across recent software engineering research is that the most consequential failures in complex software pipelines are not task-level errors — wrong answers, incorrect transformations, or bad predictions — but structural defects: gaps in component coverage, mismatches between artifact representations, and integration failures that mask the signal task-level monitors are designed to detect. This paper synthesises five corpus findings spanning agentic system monitoring, software bill of materials (SBOM) generation, binary decompilation evaluation, credential leakage detection, and multi-agent LLM collaboration topology to argue a single defensible thesis: **the dominant failure mode in modern software pipelines is structural ambiguity, not functional error, and current tooling is systematically blind to this distinction.**

The evidence base draws from cs.SE preprints covering monitoring methodology [arXiv:2606.02494](#), SBOM component inclusion [arXiv:2606.02442](#), decompilation reusability metrics [arXiv:2605.29490](#), credential classification [arXiv:2605.31520](#), and multi-agent topology experiments [arXiv:2606.01490](#). Across these domains, a consistent pattern emerges: pipelines optimised for one quality axis (readability, task accuracy, alert precision) systematically underperform on orthogonal axes (functionality, structural coverage, integration completeness), and this orthogonality is rarely measured.

Important framing note: The thesis is a *heuristic reading* of five independent studies, not a derivation from a shared formal structure. The five domains use different formalisms, different evaluation methodologies, and different operationalisations of “structural defect.” The claim is that a common pattern is visible across them — not that they are instances of a single proven theorem.

The primary falsification path is empirical: if a tool or monitoring regime can be shown to simultaneously maximise performance on all quality dimensions without explicit cross-axis measurement, the thesis fails. Current evidence suggests this has not been demonstrated for any of the reviewed systems. Secondary falsification requires showing that structural monitors add no predictive power beyond task-level monitors in a controlled regression study. The paper concludes with a proposal for a maturity-staged, multi-axis evaluation discipline applicable across these domains.

Introduction

Software engineering research has long distinguished between correctness (does the programme do what it should?) and reliability (does it consistently do so under realistic conditions?). What the recent corpus reveals — or, more precisely, what a heuristic reading of the recent corpus suggests — is a third, underexamined dimension: **structural completeness** — whether the artifact, pipeline, or tool even covers the space it claims to cover.

This paper does not derive that claim from a shared formal structure across the five studies reviewed. The studies use heterogeneous formalisms and evaluation methodologies. What it does is identify a pattern that appears across them and argue that the pattern is coherent enough to motivate a unified evaluation discipline. Readers should treat the synthesis as a candidate framework for organising future empirical work, not as a proven result.

Consider three surface-distinct problems. First, an agentic document-processing system that injects controlled task-level errors finds those errors statistically indistinguishable from clean base-lines — not because the errors are small, but because structural defects in the pipeline produce variance so large that task-level signal is swamped [arXiv:2606.02494](#). Second, SBOM generation tools across five popular implementations fail to agree on which components belong in a software bill of materials, producing “ambiguity and blind spots in component inclusion” that no individual tool resolves [arXiv:2606.02442](#). Third, binary decompilation evaluation reveals a “reusability cliff” where the best decompiler-LLM pair reaches 22.3% programme-level behavioural overlap but only 1.2% exact output match — a 20-point gap that syntactic similarity metrics would entirely miss [arXiv:2605.29490](#).

These are not coincidentally similar. They share a surface pattern: a pipeline produces an artifact, the artifact is evaluated on a single axis, and the single-axis evaluation is systematically optimistic about dimensions that were never measured. Whether this surface pattern reflects a single underlying mechanism is a theoretical question this synthesis cannot resolve — the Limitations section addresses this directly.

The motivating question for this paper is: **what evaluation discipline would make this failure pattern visible across domains?** The answer, we argue, requires decomposing pipeline evaluation into orthogonal quality dimensions, instrumenting at multiple scopes (within-run, cross-run, structural), and treating variance — not mean performance — as the primary diagnostic signal. This is not a claim about any specific tool being bad; it is a claim about the evaluation frameworks those tools are embedded in being incomplete.

Selection Process

The corpus for this synthesis was assembled from arXiv preprints submitted to cs.SE, cs.PL, and cs.CR during a 30-day window in May–June 2025. The initial candidate set comprised approximately 40 preprints identified by keyword search across the following clusters: agentic pipeline monitoring, SBOM generation and evaluation, binary decompilation and LLM-assisted repair, credential and secret detection, and multi-agent LLM collaboration.

Inclusion criteria: A paper was retained if (a) it reported empirical findings on pipeline evaluation methodology, artifact quality measurement, or structural coverage gaps; and (b) its findings were interpretable without access to the full text (i.e., the abstract and, where available, the results section provided sufficient detail to characterise the finding).

Exclusion criteria: Papers were excluded if they (a) reported only system descriptions without evaluation results; (b) addressed domains with no clear connection to software artifact quality (e.g., pure NLP benchmarks, hardware design); or (c) required specialised domain knowledge to interpret that the synthesis team did not possess.

What was filtered: Approximately 25 papers were excluded under these criteria. Notable exclusions include several preprints on LLM code generation benchmarks (excluded because they focus on task-level correctness rather than structural evaluation), and two preprints on formal verification (excluded because their evaluation methodology is not comparable to the empirical pipeline studies retained).

What was kept and why: Five primary papers were retained as the main evidence base [corpus:arxiv:2606.02494, corpus:arxiv:2606.02442, corpus:arxiv:2605.29490, corpus:arxiv:2605.31520, corpus:arxiv:2606.01490]. One additional paper [arXiv:2605.27332](#) was retained for the structural-prior finding in §3.5. One paper [arXiv:2605.27328](#) was considered for inclusion in the main argument but retained only in a weakly-connected addendum (see §A.1) because its connection to the thesis is analogical rather than evidential. Two papers on categorical semantics [corpus:arxiv:2605.31389, corpus:arxiv:2605.21337] were identified in the corpus but excluded from the main synthesis because their foundational concerns do not map onto the structural defect framework without speculative bridging.

Acknowledged limitation: This selection process was conducted by the synthesis team without external validation. The 30-day window is a convenience boundary, not a principled one. Papers that would disconfirm the structural-defects-dominate thesis may exist outside this window.

Background

2.1 The Single-Axis Evaluation Trap

Evaluation frameworks in software engineering tend to crystallise around the metrics that are easiest to compute. For decompilation, syntactic similarity (BLEU scores, edit distance) is tractable and well-understood; behavioural equivalence requires executing both the original and decompiled code under matched inputs, which is expensive [arXiv:2605.29490](#). For SBOM generation, listing which components a tool *does* identify is straightforward; characterising the space of components it *should* identify requires a ground-truth taxonomy of Component Inclusion Mechanisms (CIMs) that did not previously exist [arXiv:2606.02442](#). For credential detection, binary classification (secret vs. not-secret) is simpler to implement than three-class classification that distinguishes genuine credentials

from placeholders — and the simpler formulation produces the high false-positive rates that make existing tools operationally unreliable [arXiv:2605.31520](#).

The pattern is consistent: the evaluation axis that is easiest to measure is adopted, and the axes that require additional instrumentation are deferred or ignored. The consequence is that tools optimised against the measured axis may actively underperform on unmeasured axes — a phenomenon the decompilation study makes explicit by showing that compiler settings maximising readability minimise functionality, and vice versa [arXiv:2605.29490](#).

2.2 Structural vs. Task-Level Failure Modes

The monitoring literature distinguishes two failure regimes that are operationally distinct but often conflated. Task-level failures are errors in the output of a specific operation: a wrong classification, an incorrect transformation, a missed credential. Structural failures are defects in the pipeline itself: an integration gap between stages, a component that is never invoked, a coverage hole in the artifact representation.

The agentic monitoring study — which the authors describe as “Stage 1 evidence” on a synthetic testbed — formalises this distinction with a concrete measurement: within-run monitors surface deterministic stage defects with coefficient of variation $CV = 0.02$, cross-run monitors surface stochastic integration consequences with $CV = 1.25$, and injected task-level errors produce CV indistinguishable from clean baselines [arXiv:2606.02494](#). This is not a subtle statistical effect — the structural monitor identifies an integration gap with $CV = 0.00$ (perfect consistency across 220 runs). The implication, within the scope of this synthetic testbed, is that deploying only task-level monitors in a structurally defective pipeline produces confident but potentially meaningless evaluations. Whether this implication generalises beyond the document-processing domain is an open empirical question.

2.3 Artifact Lifecycle and Governance

A third background concept relevant to this synthesis is the distinction between artifacts as outputs and artifacts as persistent operational entities. The conventional model treats a pipeline artifact (a compiled binary, a generated SBOM, an agent-produced document) as a point-in-time output evaluated at delivery. One proposed alternative, described in a preprint on governed runtime evolution, treats artifacts as persistent runtime capabilities subject to lifecycle management, validation, and rollback [arXiv:2605.27328](#).

This distinction is potentially relevant to multi-axis evaluation because point-in-time evaluation captures only the artifact’s state at one moment in a potentially evolving lifecycle. However, the connection between this framework and the structural defect thesis is analogical rather than evidential — the governed runtime evolution paper addresses artifact mutation governance, not evaluation methodology. Readers interested in this connection are directed to the weakly-connected addendum (§A.1), where the paper is discussed with appropriate hedging.

Synthesis

3.1 Variance as a Diagnostic Signal, Not a Noise Floor

Claim: Coefficient of variation (CV) across repeated runs is a more informative diagnostic signal than mean performance for identifying structural defects in document-processing agentic pipelines, because structural defects produce characteristic variance signatures that task-level metrics cannot replicate in the studied setting.

Evidence: The agentic monitoring study directly measures this on a synthetic testbed with controlled error injection across 120 document bundles and 220 runs. Within-run monitors on deterministic stage defects produce $CV = 0.02$ — low variance, high consistency, indicating a fixed structural property. Cross-run monitors on stochastic integration consequences produce $CV = 1.25$ — high variance, indicating a non-deterministic integration boundary. Critically, injected task-level errors produce CV indistinguishable from the clean baseline, confirming that structural defects mask task-level signal rather than amplifying it [arXiv:2606.02494](https://arxiv.org/abs/2606.02494). The structural monitor achieves $CV = 0.00$, identifying an integration gap with perfect consistency.

Why this might generalise (and why it might not): The CV-as-diagnostic-signal claim rests on the observation that structural defects are, by definition, properties of the pipeline architecture rather than of individual runs — and architectural properties should produce consistent signatures across runs. This reasoning would apply to any pipeline with a fixed structural defect, not just document-processing pipelines. However, the mechanism is not formally derived: it is a plausible inference from the observed data. If structural defects in other domains are stochastic rather than deterministic (e.g., a probabilistic integration boundary), the $CV = 0.00$ signature would not appear, and the diagnostic value of CV would be reduced. The authors explicitly describe this as Stage 1 evidence.

Caveat: The CV signatures observed are specific to the document-processing domain and to the particular structural defects injected. The study does not test pipelines with stochastic structural defects, nor does it compare CV diagnostics against alternative statistical signatures (e.g., inter-quartile range, tail behaviour).

Falsification path: Run the same monitoring regime on a pipeline where structural defects are stochastic (e.g., a probabilistic integration boundary that activates on 50% of runs). If CV for the structural monitor rises above the task-level CV in that setting, the claim that $CV = 0.00$ is a reliable structural signature fails. Independently: construct a task-level error that produces $CV > 1.0$ in a structurally sound pipeline across at least 100 runs; if such errors exist, the claim that high CV is diagnostic of structural (not task-level) failure is falsified. Both tests should be conducted on pipelines outside the document-processing domain to assess generality.

3.2 Component Coverage Gaps Are Systematic, Not Random

Claim: Across SBOM generation tools, coverage gaps in component inclusion are not random sampling failures but systematic blind spots tied to which Component Inclusion Mechanisms (CIMs) each tool’s architecture recognises — and no current tool covers all CIMs.

Evidence: The SBOM analysis systematically evaluates four popular tools (cdxgen, syft, trivy, ORT) plus the Microsoft sbom-tool against a ground-truth CIM taxonomy across six programming languages (Python, Java, Go, PHP, Rust, C). The finding is not that tools perform poorly on average, but that “no tool covers all identified CIMs and that common gaps exist across tools”

[arXiv:2606.02442](#). The gaps appear architectural: tools built around package-manager manifests miss dynamically linked components; tools built around binary scanning miss source-level dependencies. These are not random misses — they are structural blind spots determined by the tool’s design assumptions about what a “component” is.

Caveat: The ground-truth CIM taxonomy is itself a construct of this study. If the taxonomy is incomplete or if reasonable alternative taxonomies would classify some “gaps” as intentional scope exclusions, the severity of the finding changes. The study does not report inter-rater reliability for the ground-truth classification, which limits confidence in the taxonomy’s objectivity.

Falsification path: Construct a CIM taxonomy through a process with documented inter-rater agreement (e.g., 0.7 across at least three independent domain experts), then re-evaluate the same five tools against it. If tool coverage gaps disappear or become randomly distributed under the independently constructed taxonomy, the claim that gaps are systematic and architectural is weakened. If gaps persist across multiple independent taxonomies with high inter-rater agreement, the claim is strengthened. A secondary test: evaluate the same tools on a new set of projects not used in the original study; if gap patterns change substantially, the finding may be project-specific rather than architectural.

3.3 Quality Axes Are Orthogonal, Not Correlated — And Optimisation Along One Axis Degrades Others

Claim: In the studied software transformation pipelines, quality dimensions (readability, recompileability, functionality; or precision, recall, false-positive rate) appear to be genuinely orthogonal axes where optimisation along one axis actively degrades performance on others — at least within the datasets and configurations evaluated.

Evidence: The decompilation study — evaluated on 240 atomic test functions across five decompilers and three repair LLMs — makes this explicit with two concrete findings. First, “-O3 yields the lowest readability but the highest functionality, and Clang gives lower readability than GCC but 2.6x higher functionality” [arXiv:2605.29490](#). These are not marginal differences — 2.6x is a large effect within this benchmark. Second, “cross-decompiler variation at the functional level is 20x, far larger than the 1.6x cross-LLM variation,” showing that the choice of decompiler (a structural choice about the pipeline architecture) dominates the choice of repair LLM (a task-level choice) by an order of magnitude in this setting.

The credential detection study provides a parallel finding on a dataset of 9,426 samples: binary classification (the simpler axis) produces high false-positive rates that make tools operationally unreliable. Adding a third class (placeholder/weak credentials) improves the operationally relevant metric (reducing high-severity alerts by 33%, from 373 to 250) without sacrificing recall on genuine leaks [arXiv:2605.31520](#). The two-class framing was not just simpler — it was optimising the wrong axis.

Why the analogy between decompilation and credential detection holds (and where it may not): Both cases involve a measurement framework that collapses a multi-dimensional quality space into a single axis, and both show that the collapsed axis is misleading about performance on the uncollapsed dimensions. The mechanism is the same: information is lost when a multi-dimensional space is projected onto a single axis. However, the *nature* of the orthogonality differs: in decompilation, the axes are in tension because compiler optimisations make different trade-offs; in credential detection, the axes are in tension because the binary label conflates distinct semantic

categories. These are different mechanisms producing a similar surface pattern — the analogy is at the level of evaluation methodology, not at the level of pipeline architecture.

Caveat: Both studies are evaluated on specific datasets. The orthogonality finding may be dataset-specific. In particular, if the test functions in the decompilation benchmark are systematically selected to maximise the readability-functionality tradeoff, the 2.6x effect may overstate the general case. The credential detection dataset’s composition (ratio of genuine credentials to placeholders) also affects the magnitude of the false-positive reduction.

Falsification path: Identify a compiler/decompiler configuration that simultaneously maximises readability and functionality in the decompilation benchmark on a held-out test set of at least 500 functions. If such a configuration exists and was not evaluated, the orthogonality claim is weakened for that benchmark. For credential detection, show that a binary classifier with optimised decision threshold can match the three-class model’s 33% alert reduction on the same dataset without the explicit placeholder class; if achievable through threshold tuning alone, the structural necessity of the third class is in question.

3.4 Topology Determines Output Quality More Than Model Capability in Multi-Agent Systems

Claim: In the studied multi-agent LLM collaboration experiment for software design, the structural topology of agent interaction explains more variance in output quality than the capability of individual models, and some topologies produce categorically lower quality regardless of model capability.

Evidence: The controlled experiment across 12 collaboration topologies and 520 runs — evaluated by three automated LLM evaluators (GPT-OSS 120B, Claude Opus 4.6, Claude Sonnet 4.6), not by human experts — finds that “parallel merge is fundamentally broken — all three evaluators place merge variants in the bottom tier (3.65-3.79), due to token starvation and the Frankenstein effect” [arXiv:2606.01490](#). The bottom-tier score (3.65-3.79 out of 5.0) versus the top-tier score (4.637) represents a gap that persists across all three automated evaluators. The structural adversarial topology (v4b) and cross-model review topology rank first and second by all evaluators, while merge variants rank last by all evaluators — the topology ranking is more consistent across evaluators than the model-specific quality rankings.

The evaluator disagreement finding reinforces this: evaluators “agree v4b is best and v3 is worst, but disagree sharply on v2b (Claude d=1.44 vs. GPT-OSS d=0.45)” [arXiv:2606.01490](#). The disagreement is on intermediate topologies, not on the structural extremes. This suggests that topology determines the ceiling and floor of achievable quality within this experiment, while model-specific factors determine position within that range — though this inference should be treated as a candidate hypothesis rather than an established finding, given the automated-evaluator limitation.

Caveat: The experiment uses automated LLM evaluators rather than human expert evaluation. If the evaluator models have systematic biases toward certain output styles (e.g., preferring adversarially generated content, or preferring outputs from models in the same family), the topology rankings may reflect evaluator bias rather than genuine design quality. The study reports cross-evaluator consistency but does not report human ground-truth validation. This is a material limitation for the strength of the topology-dominates-capability claim.

Falsification path: Replicate the experiment with human expert evaluators (software architects with 5 years of experience) on a stratified random sample of at least 60 outputs (5 per topology).

Pre-register the hypothesis that v4b ranks first and merge variants rank last. If human evaluators reverse the topology rankings — placing merge variants above adversarial variants — the claim that topology determines quality ceiling is falsified. If human evaluators agree with the automated rankings on the structural extremes but disagree on intermediates, the claim survives in its weaker form (topology determines floor and ceiling, not intermediate ordering).

3.5 Structural Priors Outperform End-to-End Learning for Topology-Critical Tasks

Claim: For the specific task of flowchart conversion from industrial documents, injecting a deterministically extracted structural prior (a Canny edge map) into a VLM’s input produces substantially better topology recovery than the same VLM without the prior — and this improvement is training-free, suggesting the prior provides information the model cannot recover from the image alone.

Evidence: The EdgeFlow study evaluates VLM-based flowchart conversion with and without a Canny edge map as a structural prior on the IndusReqFlow industrial dataset. The improvement from adding the edge map is: node-level F1 +17.39 percentage points, edge-level F1 +16.94 percentage points, path-level F1 +11.06 percentage points [arXiv:2605.27332](#). This improvement requires no annotated training data and no domain-specific fine-tuning — the structural prior is injected at inference time as an additional input channel.

The cross-dataset finding is equally important: “cross-dataset evaluation results on a public synthetic benchmark show no significant improvement” [arXiv:2605.27332](#). The structural prior helps on industrial data but not on synthetic data. One possible interpretation is that the benefit scales with structural complexity; another is that the baseline VLM is already well-calibrated for synthetic flowchart styles and the edge map adds redundant information in that setting. The abstract does not establish which interpretation is correct.

Why this might connect to the broader thesis (and why it might not): The EdgeFlow finding is consistent with the general claim that structural information not captured by task-level metrics matters for pipeline quality. However, the mechanism here is specific: a hand-crafted edge detector provides topological information that a general-purpose VLM lacks. This is a different mechanism from the CV-based structural monitoring in §3.1 or the CIM coverage gaps in §3.2. The connection is at the level of the evaluation lesson (structural information matters) rather than at the level of pipeline architecture or failure mode. Readers should not infer that edge maps are relevant to SBOM generation or agentic monitoring — the analogy is methodological, not architectural.

Caveat: The comparison is between EdgeFlow (VLM + edge map) and off-the-shelf VLMs without the edge map. It does not compare against a fine-tuned VLM trained on flowchart data, which might learn equivalent structural priors from training examples. The 17-point improvement may partly reflect the baseline VLM’s lack of domain adaptation rather than the intrinsic value of the structural prior. The industrial dataset (IndusReqFlow) is not publicly described in sufficient detail in the abstract to assess whether it is representative of industrial flowcharts generally.

Falsification path: Fine-tune a VLM of the same parameter count as the EdgeFlow baseline on the IndusReqFlow training set, without the edge map, for a training budget equivalent to the inference cost of the edge map computation across the test set. Evaluate on the same test set. If the fine-tuned model achieves node-level F1 within 3 percentage points of EdgeFlow, the claim that the structural prior provides irreducible benefit beyond what domain adaptation can provide is falsified.

The threshold of 3 percentage points is chosen as the approximate measurement uncertainty given the dataset scale; this threshold should be pre-registered before the experiment.

Discussion

What This Implies

The synthesis across these five findings points toward a discipline that might be called **multi-axis structural evaluation**: the practice of explicitly measuring pipeline quality along orthogonal dimensions, instrumenting at multiple scopes, and treating variance signatures as diagnostic signals for structural (not just task-level) defects. This is a candidate framework, not a proven methodology — the evidence base is five preprints, each with its own scope limitations.

The practical implication for SBOM tooling is that coverage gap analysis requires a CIM taxonomy developed independently of the tools being evaluated — current tools cannot self-report their own blind spots [arXiv:2606.02442](#). The practical implication for decompilation benchmarking is that any benchmark reporting only readability or only recompilability is providing information about at most one-third of the relevant quality space, within the scope of the studied benchmark [arXiv:2605.29490](#). The practical implication for agentic system deployment is that task-level monitoring should be understood as a complement to, not a substitute for, structural monitoring — and that deploying task-level monitors before structural monitors are in place may produce false confidence, at least in settings resembling the studied document-processing pipeline [arXiv:2606.02494](#).

The topology findings from multi-agent collaboration [arXiv:2606.01490](#) and flowchart conversion [arXiv:2605.27332](#) are consistent with a candidate principle: structural choices (how agents are connected, what priors are injected) may dominate capability choices (which model is used, how much training data is available) for tasks where topology is load-bearing. This is a hypothesis to be tested, not an established result. The falsification paths in §3.4 and §3.5 are the appropriate next steps. The cross-domain analogy between these two findings is at the level of evaluation lesson, not mechanism — the token-starvation failure mode in multi-agent merge topologies is a different phenomenon from the edge-detection prior in flowchart conversion, and conflating them would be a category error.

What This Does NOT Imply

This synthesis does not imply that task-level evaluation is useless. The credential detection study demonstrates that task-level metrics (recall on genuine credentials, precision on alerts) are necessary inputs to the evaluation framework — the three-class model is validated precisely because it improves on task-level metrics while adding structural discrimination [arXiv:2605.31520](#). The claim is that task-level metrics are insufficient, not that they are irrelevant.

This synthesis does not imply that structural monitors are always available or cheap to construct. The SBOM study’s ground-truth CIM taxonomy required substantial domain expertise to build [arXiv:2606.02442](#). The agentic monitoring study’s structural monitor required a testbed of 220 runs with controlled error injection [arXiv:2606.02494](#). In practice, the cost of building structural monitors may exceed the cost of the tools being evaluated — this is a real barrier that the synthesis does not resolve.

This synthesis does not imply that the findings generalise across all software domains. Each study

is bounded by its dataset, its domain, and its evaluation methodology. The decompilation findings are specific to five decompilers, three repair LLMs, and 240 atomic test functions [arXiv:2605.29490](#). The topology findings are specific to 12 collaboration topologies, 8 design tasks, and automated LLM evaluators [arXiv:2606.01490](#). Generalisation claims require replication across domains, which none of these studies provide.

Finally, this synthesis does not endorse any specific tool, topology, or architectural choice. The structural adversarial topology ranking first in the multi-agent experiment [arXiv:2606.01490](#) does not mean adversarial prompting is universally beneficial — the finding is domain-specific to software architecture design tasks evaluated by LLM judges, and the automated-evaluator caveat applies throughout.

Limitations

Preprint status: All corpus sources are arXiv preprints, not peer-reviewed publications. The findings have not been subject to external scrutiny. In particular, the multi-agent topology experiment [arXiv:2606.01490](#) relies on automated LLM evaluators rather than human ground truth, and the agentic monitoring study [arXiv:2606.02494](#) is explicitly self-described as “Stage 1 evidence.” These are not disqualifying, but they require epistemic caution throughout — including in the synthesis claims that draw on them.

Dataset scale: Several studies operate at scales that may not support strong generalisation claims. The decompilation benchmark contains 240 atomic test functions [arXiv:2605.29490](#). The credential detection dataset contains 9,426 samples [arXiv:2605.31520](#). The agentic monitoring study covers 220 runs across 120 document bundles [arXiv:2606.02494](#). These are reasonable for exploratory studies but insufficient for strong distributional claims.

Mechanism underspecification: The synthesis identifies a pattern (structural defects dominate task-level errors) but does not fully specify the mechanism by which structural defects mask task-level signal. The agentic monitoring study provides a variance-based account [arXiv:2606.02494](#), but whether this account generalises to SBOM gaps or decompilation reusability cliffs requires theoretical work not present in the corpus. The cross-domain analogy is at the level of vocabulary, not mechanism.

Construct validity: The key constructs — “structural defect,” “task-level error,” “quality axis” — are operationalised differently across studies. The agentic monitoring study operationalises structural defects as integration gaps between pipeline stages [arXiv:2606.02494](#). The SBOM study operationalises them as CIM coverage gaps [arXiv:2606.02442](#). The decompilation study operationalises them as the gap between readability and functionality axes [arXiv:2605.29490](#). Whether these are instances of the same underlying construct or merely analogically similar phenomena is a theoretical question this synthesis cannot resolve.

Selection bias: The corpus is limited to papers from a 30-day window assembled by the synthesis team without external validation. The selection process is described in the Selection Process section, but the 30-day window is a convenience boundary. Papers that would disconfirm the structural-defects-dominate thesis may exist outside this window or may have been excluded under the stated criteria.

Heuristic reading: As stated in the Introduction and Abstract, the thesis is a heuristic reading of

five independent studies, not a derivation from a shared formal structure. The pattern identified is real in the sense that it appears across the reviewed corpus; whether it reflects a single underlying phenomenon is an open question.

Conclusion

Across agentic system monitoring, SBOM generation, binary decompilation evaluation, credential leakage detection, and multi-agent collaboration topology, a consistent surface pattern emerges: the failure modes that matter most appear to be structural gaps — integration defects, component coverage blind spots, and quality-axis orthogonalities — rather than task-level errors, and current evaluation frameworks are systematically blind to these gaps because they instrument only the axis that is easiest to measure. This is a heuristic reading of five independent preprints, not a derived theorem, and the constructs are operationalised differently across studies.

The evidence from the reviewed corpus suggests — tentatively, given preprint status and dataset scale — that variance signatures (not mean performance) are a promising diagnostic signal for structural defects in the studied settings, that topology and structural priors appear to dominate model capability for topology-critical tasks in the studied experiments, and that multi-axis evaluation is necessary but currently absent from standard practice across these domains. Each of these claims carries a concrete falsification path: if controlled experiments demonstrate that task-level monitors are sufficient, that quality axes are correlated rather than orthogonal, or that structural priors add no information beyond what fine-tuning captures, the thesis fails — and the field will be better for knowing it.

Appendix: Weakly-Connected Sources

A.1 Governed Runtime Evolution [arXiv:2605.27328](#)

This paper proposes a framework for treating software artifacts as persistent runtime capabilities subject to lifecycle management, validation, and rollback — operationalised through a HarnessMutation mechanism. The connection to the structural defect thesis is analogical: if artifacts evolve through mutation, then single-point evaluation at deployment may be optimistic about long-run quality, and lifecycle-aware multi-axis evaluation would be preferable to single-point evaluation.

This connection is not evidential. The governed runtime evolution paper does not evaluate multi-axis quality metrics, does not study structural defects in the sense used by the other corpus papers, and uses a different formalism (runtime governance constraints) from the monitoring, SBOM, decompilation, and credential detection studies. The analogy is at the level of vocabulary (“lifecycle,” “validation”) rather than mechanism.

The paper is retained here rather than in the main synthesis because (a) the citation exists in the corpus and should remain traceable, and (b) the lifecycle-aware evaluation concept is a plausible extension of the multi-axis evaluation discipline proposed in this paper — but it requires empirical work not present in the corpus to establish the connection. Readers interested in this extension should consult the paper directly and treat any connection to the structural defect thesis as speculative.

Response to Review

Heuristic-vs-derivational gap (addressed): The Introduction and Abstract now explicitly name the thesis as a “heuristic reading” of five independent studies rather than a derivation from a shared formal structure. The phrase “candidate framework” is used throughout the Discussion to maintain this framing. The original language implied more formal unity than the evidence supports.

Undisclosed selection (addressed): A new “Selection Process” section has been added between the Introduction and Background. It names the initial candidate set (~40 papers), the inclusion and exclusion criteria, what was filtered and why, and what was kept and why. The two categorical semantics papers are explicitly accounted for. The 30-day window is acknowledged as a convenience boundary rather than a principled one.

Weakest-link source (addressed): The governed runtime evolution paper [arXiv:2605.27328](#) has been removed from the main argument (§2.3 has been substantially revised to flag the analogical nature of the connection and redirect readers to the appendix) and relocated to a new “Appendix: Weakly-Connected Sources” section (§A.1). The citation is preserved and traceable; the connection is now explicitly characterised as analogical rather than evidential.

Abstract-only overshoot and dropped caveats (addressed): The “Stage 1 evidence” caveat from the agentic monitoring study is now restored at point of use in §2.2 and §3.1, not only in the Limitations. The decompilation study’s scope (240 atomic test functions, specific decompiler/LLM combinations) is now noted in §3.3 at point of use. The multi-agent study’s automated-evaluator limitation is now noted in §3.4 at point of use, and the topology-dominates-capability claim is downgraded to “candidate hypothesis.” The EdgeFlow cross-dataset finding is now presented with two possible interpretations rather than a single causal inference.

Asserted-not-argued cross-domain analogies (addressed): §3.3 now includes an explicit paragraph arguing *why* the decompilation and credential detection findings are analogous (both involve collapsing a multi-dimensional quality space onto a single axis, losing information) and *where the analogy breaks down* (different mechanisms produce the orthogonality). §3.5 now explicitly states that the connection to the broader thesis is at the level of evaluation lesson, not pipeline architecture, and warns against conflating the edge-map mechanism with the token-starvation mechanism in §3.4. The Discussion’s “convergent principle” language has been replaced with “candidate principle to be tested.”

Weak or absent falsification paths (addressed): The falsification path for §3.1 now specifies a minimum run count (100 runs), a specific alternative pipeline type (probabilistic integration boundary), and a domain-general requirement. The falsification path for §3.2 now specifies a minimum inter-rater agreement threshold (> 0.7 , three independent experts) and a secondary replication test on new projects. The falsification path for §3.5 now specifies a pre-registration requirement, a training budget equivalence condition, and a 3-percentage-point threshold with explicit justification. The Discussion’s “convergent principle” claim now has an explicit pointer to §3.4 and §3.5 as the appropriate falsification loci rather than standing as an unanchored assertion.

Historiographic overreach (addressed): The phrase “an emerging alternative... treats artifacts as persistent runtime capabilities” in §2.3 has been revised to “one proposed alternative” and the section now redirects to the appendix rather than presenting the framework as a field-level development. No “paradigm shift” language was present in the original, but the §2.3 framing implied broader field adoption than a single preprint supports.