Ghost Intent

An Effect of Traceability Collapse in GenAI-Assisted SDLCs

Author: Spark Tsai
ORCID: https://orcid.org/0009-0006-8847-4703 **Email:** spark.tsai@gmail.com **Date:** February 2026

Abstract

As GenAI-assisted software development shifts the human role from authorship to review-centric oversight, foundational assumptions of software traceability are systematically failing. Engineers increasingly encounter code that is **behaviorally coherent**, readable, and previously reviewed, yet lacks any recoverable design rationale or navigable entry point for debugging and modification.

This paper defines **Ghost Intent** as the experiential manifestation of **Authorless Traceability Collapse (ATC)**—a structural condition in which engineering intent evaporates after generation, leaving only behavioral remnants embedded in artifacts. We argue that Ghost Intent invalidates long-standing SDLC assumptions and produces a **debug cost inversion**, where generation accelerates while comprehension and diagnostic effort expand non-linearly.

Rather than proposing tools or workflows, this paper positions Ghost Intent as a first-order governance risk in GenAI-assisted development. By articulating its formal characteristics, observability signals, and multi-dimensional risk profile, we establish a shared conceptual vocabulary for future AI-native SDLC formalization and decision behavior governance.

Keywords

Ghost Intent, Authorless Traceability Collapse (ATC), Intent Evaporation, GenAI-Assisted Software Development, AI-Native SDLC, Debug Cost Inversion, Decision Provenance, Traceability Assumption Failure, Governance Risk, Review-Centric Development

Introduction

Generative AI has fundamentally altered the mechanics of software development. Code is increasingly generated by models and integrated through human review rather than authored through sustained human reasoning. While this shift appears to accelerate delivery, it introduces a new class of instability in how systems are understood, debugged, and governed over time.

Practitioners report a recurring paradox: defects arise in code they previously reviewed and approved, yet no diagnostic entry point exists to understand *why* the code was structured this way. Version control is intact, syntax is readable, and runtime behavior is observable—but the decision context that justified the implementation cannot be recovered. Debugging shifts from correction to **interpretive archaeology**, where engineers reconstruct intent through behavioral inference rather than deductive reasoning.

These experiences are not anecdotal failures or skill gaps. They reflect a structural breakdown in foundational traceability assumptions under authorless, review-centric development conditions.

Background and Motivation

Traditional SDLC models implicitly assume durable human authorship. Traceability artifacts—commit histories, pull request discussions, and documentation—presume that intent is either encoded in artifacts or retained in human memory.

GenAI-assisted workflows violate this assumption. Code may be generated without a stable human mental model ever forming. Review validates correctness at a moment in time but does not preserve the decision logic that shaped architectural boundaries, constraints, or trade-offs.

As a result, organizations accumulate systems that are functionally correct but **epistemically opaque**. This gap between preserved artifacts and evaporated intent necessitates a conceptual reframing that goes beyond traditional traceability models.

Defining Ghost Intent

Ghost Intent characterizes a condition in which software artifacts remain operational and behaviorally coherent, yet the engineering decisions that justified their structure, boundaries, or behavior are no longer recoverable.

Unlike undocumented or legacy code—which suffer from post-hoc knowledge decay—Ghost Intent arises from **absent-from-inception** decision capture. It emerges when AI-assisted generation produces artifacts without ever forming a durable association between code and the decision units, constraints, or alternatives that shaped them.

Ghost Intent is neither a defect nor a documentation gap. It is a **structural decoupling between behavioral correctness and decision provenance**, made visible primarily during debugging, refactoring, or system evolution.

Minimal Formula Representation

Ghost Intent can be minimally formalized as a discrepancy between preserved artifacts and absent decision records. Let:

- $D(a)$ denote the set of engineering decisions originally associated with artifact a (constraints, boundary choices, alternatives, confidence assumptions)
- $A(a)$ denote the observable artifacts produced (code, tests, configuration)

Ghost Intent arises when:

$$ I_{lost}(a) = D(a) - A(a) \neq \varnothing $$

In traditional SDLCs, teams often infer $D(a)$ from $A(a)$ through authorship and historical context. Under Authorless Traceability Collapse, this inference becomes **non-injective**: multiple incompatible decision paths map to identical artifacts, rendering reverse reconstruction epistemically unreliable.

The inference from $A(a)$ to $D(a)$ thus represents a **lossy decompression of engineering intent**, where stochastic generation introduces irreversible decision entropy into the codebase.

Intent Anchor

An **Intent Anchor** denotes any stable, navigable reference that binds a design or implementation decision to a specific location within a software system.

In traditional development, intent anchors emerge implicitly through authorship, architectural conventions, or long-lived discussion artifacts. In GenAI-assisted pipelines, such anchors frequently fail to materialize. Generated code may be reviewed and merged without persistent linkage to prompt context, boundary assumptions, or discarded alternatives.

The concept of Intent Anchor draws from related discussions on decision boundaries and traceability loss in AI-assisted pipelines (Tsai, 2026). Its absence marks an **anchor-less state**, in which artifacts persist without recoverable decision coordinates.

Authorless Traceability Collapse

Authorless Traceability Collapse (ATC) describes the systematic failure of traditional traceability mechanisms under conditions where software is primarily generated by AI and integrated through human review rather than authored.

Conventional traceability records *what* changed and *that* approval occurred, but presumes the persistence of an accountable author who can explain *why*. When authorship is displaced, traceability artifacts lose their explanatory power.

ATC is structural, not procedural. Existing tools function as designed, but their foundational assumptions no longer hold.

Observability of Ghost Intent

Ghost Intent is defined by the absence of decision evidence and therefore cannot be directly logged or inspected. It is observable only through indirect signals:

- Elevated debug-to-generation time ratios
- Absence of a reliable diagnostic entry point
- Semantic search yielding behavior patterns but no explanatory context
- Regressions introduced by seemingly minor modifications

These signals do not recover lost intent. They enable recognition of Ghost Intent as a structural condition distinct from routine debugging challenges.

Structured Breakdown of SDLC Assumption Failure

Traditional SDLC models assume that intent is durable, attributable, and recoverable across lifecycle stages. In GenAI-assisted development, generation outpaces cognition. Code may be produced without stable requirements, merged without durable rationale, and modified without awareness of prior constraints.

The mapping from intent to artifact becomes discontinuous.

Observed Consequences

Productivity Metrics Failure

Velocity-based metrics capture generation speed while obscuring downstream comprehension and rework costs, creating a productivity illusion.

Debug Cost Inversion

Debugging effort frequently exceeds generation effort. Engineers shift from deductive reasoning to inductive inference, engaging in interpretive archaeology rather than correction.

Deferred and Amplified Risk

Risk compounds through scale-based, iteration-based, and time-deferred amplification.

Structural Risk from Deletion and External Integration

Without preserved intent, deletion and integration operations acquire asymmetric risk. Invisible coupling emerges from undocumented assumptions embedded in generated code.

Risk Classification of Ghost Intent

Dimension	Risk Manifestation
Artifact Alignment	Intent–artifact misalignment, version drift, orphaned intent
SDLC Lifecycle	Generation ambiguity, review-phase decision loss, maintenance entry collapse
Amplification Mechanism	Scale-based, iteration-based, time-deferred amplification
Impact Scope	Maintainability, productivity, governance, organizational knowledge
Cognitive & Decision	Accountability fog, cognitive entry-point collapse, decision vacancy

Implications for SDLC and Governance

Ghost Intent reveals a governance blind spot: systems may be behaviorally correct while becoming progressively incomprehensible to their stewards. This shifts failure analysis from code-centric defects to **decision-centric risk**.

Addressing Ghost Intent requires reorienting SDLC models from artifact pipelines toward decision-preserving lifecycles.

Scope and Non-Goals

This paper does not propose tools, enforcement mechanisms, or automated capture systems. Its purpose is to define Ghost Intent, articulate its risks, and establish conceptual foundations for subsequent formalization.

Conclusion

Ghost Intent captures a new failure mode in GenAI-assisted software development: the persistence of behavior without recoverable intent.

This paper does not argue that GenAI slows development, but that it invalidates the assumptions by which development speed has traditionally been measured and governed.

Related Work

Empirical studies document a productivity paradox in GenAI-assisted software engineering, where apparent acceleration masks increased verification and maintenance costs. Parallel research on code provenance and explainability highlights persistent gaps in decision traceability. Governance frameworks from NIST and OECD emphasize decision accountability and auditability as critical AI risk dimensions.

This work provides a unifying conceptual lens connecting these observations to SDLC-level governance failure.

Disclosure of AI Assistance

Generative AI tools were used selectively to assist with language refinement and editorial clarity. All conceptual frameworks, formal definitions, and analytical structures presented in this paper were developed and validated by the author.

References

- Tsai, S. (2026). *From Inference Creep to Risk Acceleration Pipelines*. SSRN Working Paper No. 6146686.
- Developer Productivity with GenAI. (2025). *arXiv:2510.24265*. https://arxiv.org/abs/2510.24265
- An Empirical Study of Generative AI Adoption in Software Engineering. (2025). *arXiv:2512.23327*. https://arxiv.org/abs/2512.23327
- Explainability as a Compliance Requirement: What Regulated Industries Need from AI Tools for Design Artifact Generation. (2025). *arXiv:2507.09220*. https://arxiv.org/abs/2507.09220
- NIST. (2025). *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile (NIST AI 600-1)*. https://doi.org/10.6028/NIST.AI.600-1
- OECD. (2022/2025). *Framework for the Classification of AI Systems*. https://www.oecd.org/
- Will It Survive? Deciphering the Fate of AI-Generated Code in Open Source. (2026). *arXiv:2601.16809*. https://arxiv.org/abs/2601.16809
- Stack Overflow. (2025). *Developer Survey: AI Coding Tools*.
- METR. (2024). *Measuring the Productivity Impact of AI-Assisted Software Development*.
- Harness. (2025). *State of Software Delivery*.
- GitClear. (2025). *AI Copilot Code Quality Report*.
- Watson, E. A., & Van Itallie, M. (2025). *The Transparency Imperative in Generative AI Codebases*. SSRN.