

Polycentric Generative-Assurance Theory: Toward Adaptive Governance in Generative AI-Augmented Software Assurance

Morteza Safaei Pour
San Diego State University
msafaeipour@sdsu.edu

Kaveh Abhari
San Diego State University
kabhari@sdsu.edu

Farzad Fathi
Bucknell University
ff009@bucknell.edu

Abstract

The integration of generative AI (GenAI) into software development is transforming how code is authored, reviewed, and assured. While GenAI boosts productivity and creativity, it disrupts longstanding assurance frameworks, introducing epistemic opacity, validation deficits, accountability ambiguities, and governance challenges. This paper introduces Polycentric Generative-Assurance Theory (PGAT), a sociotechnical framework explaining how trust in AI-generated code is sustained through five interdependent responsibilities: epistemic mapping, adversarial socio-technical analysis, meta-validation, computational ethics, and evolutionary governance. Our findings reveal that assurance is no longer linear or role-bound, but rather a distributed, adaptive, and emergent practice. PGAT reframes assurance as a responsible process of trust orchestration, where multiple responsibilities coalesce to ensure the reliability, maintainability, and ethical integrity of software development practices.

Keywords: Generative AI, software development, AI-assisted coding, information assurance, responsibility, polycentric generative-assurance theory.

1 Introduction

The integration of generative AI (GenAI) into software development marks more than a technological evolution—it constitutes a paradigmatic shift in how code is created, evaluated, and assured. While GenAI promises significant gains in productivity and creative acceleration, it also introduces deep structural challenges that unsettle long-established norms of software quality, security, and professional accountability (Negi et al., 2025; Tabarsi et al., 2025). Modern development teams operate within established assurance frameworks such as IEEE Std 1028-2008, Google’s *How to Do a Code Review* guidelines, and SAFECODE’s *Principles for Software Assurance Assessment* (IEEE, 2008; Winter & Wright, 2020; SAFECODE, 2019). These

frameworks emphasize incremental change, peer accountability, and defect detection through a layered combination of human review and automated tooling. Crucially, they assume a human-centered workflow: code authored with clear provenance, reviewed by peers with shared context, and integrated under conditions of epistemic transparency. The rise of AI pair programming upends these assumptions.

As GenAI tools become increasingly embedded in software engineering workflows, they accelerate the rate and volume of code production, but often at the expense of quality, interpretability, and governance. A 2025 analysis of over 200 million changed lines by GitClear reveals sharp increases in duplicated blocks, fewer refactorings, and elevated churn rates—trends symptomatic of what they refer to as the *boilerplate effect* (Harding, 2025). Reviewers are now confronted with large, opaque contributions that may include unfamiliar libraries, ambiguous licenses, or silent security vulnerabilities—all produced in seconds by an AI agent. These shifts stretch the cognitive and procedural limits of existing review infrastructures, which were never designed for this level of velocity, opacity, or probabilistic output.

While traditional review frameworks still serve as foundational guardrails, AI-generated code exposes their latent fragilities. It overwhelms human reviewers with scale, conceals intent and authorship, and introduces subtle risks that often escape automated checks. Telemetry data show that, since the widespread adoption of LLM-based copilots, copy-pasted lines now outnumber refactorings and duplicated code blocks have increased eight-fold—clear signs of declining maintainability (Harding, 2025). Complementary field studies have further shown that reviewer accuracy declines significantly as patch sizes grow, even in highly optimized settings (Sadowski et al., 2018). Addressing these compounding paradoxes will require governance models that go beyond conventional checklists—models capable of tracing provenance, surfacing epistemic blind spots, and allocating new human responsibilities in an AI-mediated environment.

This study thus centers on a critical question: *What new human responsibilities emerge to uphold information assurance as development teams adapt to the scale, opacity, and velocity introduced by generative AI in software development?* We argue that sustaining AI-augmented development requires more than seamless integration of coding and testing; it demands a fundamental reconfiguration of responsibilities to preserve meaningful human oversight amid pervasive AI contributions.

To that end, we introduce Polycentric Generative-Assurance Theory (PGAT)—a sociotechnical framework that explains how trust in AI-authored code emerges through five mutually reinforcing responsibilities: epistemic mapping, adversarial socio-technical analysis, meta-validation, computational ethics, and evolutionary governance. These responsibilities were inductively derived from thousands of practitioner discussions across twelve high-traffic Reddit communities focused on software development, providing the framework with ecological validity and timely relevance. As a result, this study contributes to both theory and practice: theoretically, by advancing a grounded model of assurance tailored to the unique epistemic and procedural risks of GenAI-mediated development; and practically, by identifying actionable responsibility archetypes that can be adopted for software development governance to strengthen oversight, resilience, and trust in AI-augmented workflows.

2 Research Background

Responsible AI governance is still in its infancy, with most frameworks fixated on outcomes and checkbox compliance rather than the deeper design decisions and dynamic processes that truly shape AI behavior (Rivera et al., 2025). Among potential domains, software development offers one of the most mature and practice-rich contexts for exploring responsible governance, providing well-established iterative workflows, technical standards, and an increasing integration of GenAI tools.

2.1 Software Development Augmentation

A substantial body of literature has documented and examined the role of GenAI in software development. Wang et al. (2022) introduced Themisto, a human-centered AI assistant that enhanced code documentation in computational notebooks, thereby improving efficiency and satisfaction. Sergeyyuk et al. (2025) and Prather et al. (2023) investigated the impact of GitHub Copilot on developer workflows, highlighting increased productivity while also

acknowledging the challenges of integrating AI into creative coding processes. Paradis et al. (2024) extended this line of inquiry with one of the largest randomized controlled trials, demonstrating that AI-assisted coding features can substantially accelerate development speed among professional engineers, although the effect size varies with experience and context. Similarly, controlled experiments by Weber et al. (2024) showed that LLM-based coding assistants like Copilot and GPT-3 could significantly boost developer productivity and satisfaction. The study also highlighted that interface and interaction design had a strong influence on these benefits. Complementing these findings, Negi et al. (2025) demonstrated that while GenAI can boost individual productivity, it paradoxically harms overall project outcomes in agile teams. Their simulation results highlighted that, without refinement and review, rapid code generation could lead to increased technical debt and rework.

Qualitative research with industry practitioners offered further nuance, revealing that while LLMs such as Copilot and ChatGPT streamlined routine coding tasks, developers remained responsible for carefully reviewing AI-generated code due to persistent errors (Tabarsi et al., 2025). In response, teams developed new workflows and best practices to mitigate these risks. Recent systematic reviews (e.g., Ebrahim & Tanner, 2025) consistently found that ChatGPT enhanced productivity in software development activities, including code generation and debugging, while also identifying persistent challenges in measuring productivity gains and fostering effective human–AI collaboration.

Donvir (2024) provided a comprehensive review of GenAI tools, including Copilot, Cursor AI, and Devin AI, highlighting their rapid evolution toward semi-autonomous coding. In parallel, Jackson et al. (2025) outlined a research agenda examining the impact of GenAI on creative thinking within software teams, urging a systematic investigation into its effects on ideation and collaboration. Promises and Pitfalls (2024) raised critical concerns about hallucinations, security vulnerabilities, and declining software quality, underscoring the need for robust and responsible evaluation frameworks. Stohr et al. (2024) adopted a theoretical lens to explore how GenAI facilitates predictive maintenance, while also emphasizing the imperative for interpretability and governance. Together, these studies illuminate a central paradox: GenAI expands the capacity for augmentation, yet simultaneously demands new forms of human judgment, oversight, and sensemaking to counterbalance its risks and uncertainties. This tension exemplifies the AI productivity paradox (Parteka & Kordalska, 2023; Wang et al., 2023).

2.2 Generative AI Inherent Limitations

While GenAI systems have demonstrated remarkable utility in accelerating software development workflows, they remain constrained by several inherent limitations (e.g., Bendig & Bräunche, 2024; Dwivedi et al., 2023). Chief among these is their lack of contextual understanding and epistemic grounding. Trained on vast, uncurated corpora, GenAI models frequently generate plausible but inaccurate outputs, a phenomenon widely referred to as hallucination. This undermines reliability, particularly in domains that require precision, traceability, and expert judgment. Additionally, GenAI systems often struggle with transparency and interpretability. Their outputs are shaped by probabilistic pattern-matching rather than intentional reasoning, which obscures provenance and complicates downstream validation. These systems also tend to reinforce dominant patterns in their training data, leading to homogenization and a decline in originality over time, which limits their effectiveness in ideation and novel problem-solving. These limitations are the root cause of the AI productivity paradox (Parteka & Kordalska, 2023; Wang et al., 2023); as GenAI becomes more capable and integrated into technical workflows, it simultaneously amplifies the need for human oversight, critical thinking, and adaptive responsibility to govern its blind spots and mitigate emerging risks.

2.3 Responsible Use of GenAI

In the context of software development, responsible use of GenAI is inseparable from the challenge of sustaining information assurance—the preservation of integrity, reliability, and traceability across increasingly automated and opaque workflows (Söllner et al., 2025). As GenAI tools such as Copilot and Cursor become tightly woven into daily development tasks, traditional notions of quality control, authorship, and accountability are being stretched (Söllner et al., 2025). These tools not only accelerate code generation but also complicate efforts to verify its provenance, intent, and compliance—core pillars of information assurance (Banh et al., 2025).

Existing organizational models, shaped by decades of deterministic development paradigms, remain poorly equipped to govern the probabilistic and generative logic of GenAI (Sunyaev et al., 2025). While institutional analyses have identified key adoption drivers—such as normative and mimetic pressures (Zhang et al., 2025)—and theoretical work has conceptualized GenAI as a governance-disrupting boundary resource (Mayer, 2025). These accounts

often treat responsibility as fixed or externally imposed. What remains underdeveloped is an understanding of how responsibilities for assurance are internally negotiated, dynamically enacted, and redistributed across human and machine actors.

Our study reframes the responsible use of AI as a problem of business realignment for assurance (Mucha et al., 2023). It argues that preserving trust in AI-mediated software development requires more than procedural safeguards; it necessitates that organizations deliberately reconfigure how assurance responsibilities are understood and operationalized. By surfacing these evolving practices, we move the governance conversation from compliance to capability—from documenting failure points to cultivating adaptive systems of accountability that are fit for AI-enhanced environments. This shift lays the groundwork for a governance approach centered on information assurance and responsive to the evolving demands of AI-augmented software development.

3 Methods

This study employed a data-driven, exploratory research design to investigate how software developers experienced and responded to the emergence of GenAI tools. Instead of starting with predefined constructs, we utilized large-scale online discourse to inductively derive patterns of concern, responsibility, and adaptation. Our phenomenon-driven theorization approach (Gregory & Henfridsson, 2021) enabled the development of a contextually grounded theory of adaptive governance in augmented software assurance, rooted in observed language, practices, and interactions.

Reddit was chosen as the primary data source due to its active technical communities (Vasist et al. 2025). The platform offered access to context-rich reflections from both developers who use GenAI for coding and testers who evaluate the quality, functionality, and reliability of the generated code. By integrating natural language processing, semantic clustering, and human-in-the-loop interpretation, we generated structured insights from unstructured text, connecting informal discourse to formal conceptual development.

3.1 Data Collection and Cleaning

We first compiled a comprehensive list of candidate subreddits. We retained only those that (i) explicitly centered on software engineering or GenAI (domain relevance) and (ii) maintained sizable and active memberships to ensure diverse and robust discourse. This rigorous selection yielded twelve forums, including programming, machine learning,

sysadmin, ExperiencedDevs, ChatGPTCoding, learnprogramming—whose subscriber bases range from approximately 70,000 to 7 million users. Using the Reddit API, we systematically harvested posts and comment threads on March 6, 2025. The dataset was limited to discussions initiated within the previous two years, with most originating in the prior twelve months. To ensure relevance to AI-assisted software development, we focused on threads containing keywords such as “AI coding,” “AI programming,” “Copilot,” and “LLM-generated code.” This initial scraping yielded approximately 4,992 Reddit posts and comments. The dataset was then filtered to remove unrelated content and retain only those posts explicitly engaging with information assurance concerns, forming the basis for subsequent analysis. More specifically, we applied a unified preprocessing and classification pipeline. The dataset was first cleaned by removing short or off-topic entries, URLs and HTML tags. Using OpenAI’s GPT-3.5 API, we then classified the cleaned data for relevance to information assurance in AI-assisted development. This process involved iterative prompt refinement, batch testing (100 entries at a time), and manual validation.

Information assurance relevant records ($n = 2,975$) were subsequently segmented by users’ roles, distinguishing Coders (focused on code creation and debugging) from Testers (focused on assurance, verification, review, and risk detection). Using GPT-3.5 and iterative validation, we identified 2,112 coder-related and 1,499 tester-related comments, with some overlapping both roles due to shared concerns across development and testing. Random samples (about 2%) from each classification stage were manually reviewed, and any discrepancies were used to refine prompts and improve model accuracy. This process ensured that sufficient evidence was gathered from both groups—those engaged in development and those responsible for assurance—capturing their distinct as well as overlapping perspectives.

3.2 Semantic Embedding and Clustering

To uncover deeper thematic structures within the dataset, we transformed the classified textual data into semantic vector representations. Traditional keyword-based methods often fail to capture subtle conceptual similarities across varied linguistic expressions (Jin, & Lin, 2024). In contrast, semantic embedding enables the detection of underlying meaning, allowing semantically related ideas, despite lexical variation, to be grouped effectively (Wang et al., 2023). We generated embeddings for all Tester-labeled Reddit comments using the all-MiniLM-L6-v2 model from SentenceTransformers (Reimers & Gurevych, 2019).

This model is optimized for semantic similarity tasks and is well-suited for capturing the nuanced discourse of software practitioners discussing complex information assurance challenges. Each comment was encoded as a high-dimensional vector, positioning semantically related entries in proximity within the embedding space. This representation facilitated meaningful clustering based on shared concerns and latent thematic relationships. After embedding the text, we applied agglomerative hierarchical clustering with Ward’s linkage, which successively merges clusters to minimize the increase in within-cluster variance, yielding compact and interpretable groups (Chen et al. 2025). The number-of-clusters versus linkage plot shows a distinct elbow where the rate of cluster reduction flattens (Chen et al. 2025). Cut-points around this bend were examined, and manual inspection of representative comments confirmed that a five-cluster solution best balanced granularity and coherence at the top level. Recursively applying the same criterion within each top-level branch produced a second elbow, yielding a total of 12 subclusters. These five overarching clusters and subsidiary clusters encapsulate the principal themes of challenges.

3.3 Responsibility Pattern Extraction

We conducted a second-stage analysis to model the enacted responsibilities within the twelve challenge subclusters. For each subcluster, the aggregated Reddit comment corpus was submitted to the OpenAI GPT-3.5 API. A zero-shot, structured-output prompt was engineered through iterative refinement and manual validation to extract paired practitioner actions and their articulated objectives. The resulting action-objective pairs were organized into a matrix for thematic analysis. Two independent coders performed axial coding to group semantically similar pairs into higher-order categories. Inter-coder reliability, measured using Cohen’s κ , was 0.87—exceeding the 0.80 threshold for substantial agreement. This abductive, human-in-the-loop protocol enabled the data-driven emergence of responsibility constructs, thereby bypassing predefined taxonomies (Drori & Te’eni, 2024; Kumar et al., 2024). This process surfaced five distinct responsibility archetypes, forming an empirical basis for subsequent theory development.

4 Key Findings

4.1 Information Assurance Challenges

Our empirical analysis surfaced a constellation of assurance challenges that undermine the

trustworthiness of GenAI-generated code and stretch the limits of traditional information assurance frameworks. While varied in form, these challenges cluster around five interrelated assurance domains—comprehension, accountability, validation, vulnerability, and resilience—and 12 associated risks, as identified by our topic-mining protocol (Table 1). Each domain presents distinctive obstacles that shaped the need for the five core responsibilities.

Table 1. Common Challenges

THEMES	1 ST -ORDER CHALLENGES*
Comprehension: Epistemic Uncertainty/Black- Box Comprehension	1. Black-box comprehension 9. Domain-knowledge mismatch 10. Non-deterministic re-generation
Validation: Hidden Workloads and Testing Deficiencies	2. Hidden debug-and-fix workload 3. Missing or superficial tests 4. Test-as-code quality problems
Accountability: Ambiguity in Attribution/IP	5. Copyright & license ambiguity 12. Attribution & credit issues
Vulnerability: Ethical or Security Oversights	6. Security oversights 11. Over-confidence bias
Resilience: Fragile Performance and Maintainability	7. Performance & scalability unknowns 8. Maintainability gaps

* Numbers illustrate the ranking of prevalent challenges

1. Epistemic Uncertainty and Black-Box Comprehension. One of the most pervasive challenges identified was the difficulty in understanding how GenAI-generated code arrives at specific outputs. Participants frequently described grappling with black-box comprehension (Challenge 1), where unfamiliar logic patterns and undocumented assumptions created epistemic blind spots. This issue was compounded by non-deterministic re-generation (Challenge 10), where seemingly identical prompts produced divergent outputs, undermining reproducibility. Additionally, domain-knowledge mismatches (Challenge 9) introduced further opacity, as generated code often failed to align with contextual or industry-specific standards, requiring costly manual inspection. These findings underscore the need for an epistemic mapping responsibility to make the invisible structure of generative logic interpretable.

2. Hidden Workloads and Testing Deficiencies. Beyond comprehension, participants noted significant challenges in verification and testing. Teams encountered hidden debug-and-fix workloads (Challenge 2), where surface-level correctness masked latent errors that required extensive rework. The presence of missing or superficial tests (Challenge 3) and test-as-code quality problems (Challenge 4) further compromised trust in the testing pipeline. Because generated tests often reflected the same logic

patterns as the code they were meant to validate, teams expressed concern about “testing the ghost with its own shadow.” These limitations called for a meta-validation responsibility to audit both what was tested and how testing itself was structured.

3. Ambiguity in Attribution and Intellectual Property. Assurance was also undermined by unclear boundaries of ownership and responsibility. Several individuals reported confusion regarding copyright and licensing ambiguity (Challenge 5), particularly when code snippets closely resembled publicly available training data. In parallel, attribution and credit issues (Challenge 12) surfaced in collaborative settings, where it was unclear how to assign authorship for AI-assisted contributions. These challenges eroded accountability and created ethical tensions among team members. Addressing them required a socio-technical analysis responsibility, one focused not solely on technical assurance but on anticipating the social, legal, and organizational risks associated with generative workflows.

4. Ethical and Security Oversights. Another cluster of concerns is related to oversight and underestimation of risk. Testers described instances where the authoritative tone of GenAI output led to overconfidence bias (Challenge 11), reducing critical scrutiny during code reviews. Additionally, security oversights (Challenge 6) often arose from overly generic or non-contextualized code that lacked appropriate threat modeling and adherence to best practices. These findings highlighted the need for a computational ethics framework to embed ethical and risk-aware reasoning into generative pipelines, thereby managing vulnerability.

5. Fragile Performance and Maintainability. Finally, the long-term resilience of GenAI-generated code was called into question. Participants observed performance and scalability unknowns (Challenge 7), where generated code operated well in constrained environments but faltered under production-level load. Moreover, maintainability gaps (Challenge 8) emerged because the code lacked modularity, consistency, or sufficient documentation for handoff to future developers. These limitations necessitate continuous oversight, adaptation, and protocol refinement—functions embodied in the evolutionary governance responsibility.

Together, these interlocking challenges reveal that information assurance in the generative era is no longer a matter of linear quality control. It is instead a distributed, reflexive, and co-evolving endeavor. These findings directly inform the five emerging responsibilities, each mapped to a distinct assurance domain, and collectively underscore the need for a more polycentric model of assurance.

4.2 Deriving Assurance Responsibilities

To move beyond identifying challenges and toward a generative model of assurance, we employed a two-step analytic process. First, we organized developer and tester quotes into challenge categories based on the five thematic domains: comprehension, accountability, validation, vulnerability, and resilience. This allowed us to bucket qualitative data under specific assurance challenges, such as black-box comprehension, attribution issues, or security.

Second, we used an LLM-assisted analysis (OpenAI GPT-3.5 API) to identify patterns in how participants responded to these challenges, either through their current practices or through reflective suggestions for improvement. Rather than asking what roles participants held formally, we asked: What work did they do to mitigate assurance breakdowns? What responsibilities did they assume, and how did they describe them? Through this abductive synthesis, we uncovered five recurring responsibility archetypes:

1. Epistemic Cartographer: Individuals who actively mapped out knowledge gaps, surfaced assumptions, and translated AI logic into accessible explanations. They addressed concerns about black-box comprehension, domain mismatch, and reproducibility by constructing traceable epistemic landscapes for others to navigate.

2. Adversarial Socio-Technical Analyst: Practitioners who proactively examined the social, legal, and organizational risks of code use, advocating for attribution clarity, licensing awareness, and responsible integration. These individuals recognized assurance as a socio-technical construct and treated risk anticipation as a shared team obligation.

3. Meta-Validator of Validation: Team members who critiqued and reconfigured testing logic, especially when AI-generated tests replicated code defects or superficial heuristics. They elevated validation to a second-order function, validating not only code but also the assumptions behind verification mechanisms.

4. Computational Ethicist: Contributors who embedded ethical constraints and values into the design and evaluation of generative outputs. They challenged the illusion of AI infallibility, mitigated overconfidence bias, and surfaced blind spots in fairness, security, and user impact.

5. Evolutionary Governance Architect: Those who adapted protocols, documentation standards, and deployment practices to account for emerging vulnerabilities and long-term code maintainability. These individuals viewed assurance as a dynamic process: one that requires continual rulemaking, adjustment, and reflection. These archetypes did not

align neatly with formal organizational job titles; instead, they emerged fluidly, sometimes dispersed across teams, other times concentrated in the practices of individual actors. Crucially, they reflect additional responsibilities that practitioners assume when using GenAI to generate code or reviewing GenAI-driven outputs. Their grounding in everyday practice, rather than formal policy, signals a shift from role-based authority to responsibility-centered assurance—a defining feature of resilient software development in the GenAI era. By surfacing these self-assumed assurance roles, this grounded theorization advances a relational, polycentric model of decentralized governance—one in which distributed responsibilities operate collaboratively to transform AI-generated code from a probabilistic artifact into a continuously validated sociotechnical asset. These findings highlight how governance mechanisms can build on this foundation to play a more systematic role: by recognizing, formalizing, and coordinating these emergent responsibilities.

5 Discussion

The findings of this study suggest a paradigm shift in the responsibilities and configurations of information assurance teams when engaging with GenAI-generated code. Traditional assurance roles—often linear, siloed, and task-specific—are ill-suited for the probabilistic nature of AI-generated artifacts. Our analysis reveals that assurance responsibilities are no longer monolithic nor discretely assigned. Instead, they are increasingly distributed, overlapping, and situationally enacted, often requiring a single actor to perform multiple assurance functions dynamically. This shift necessitates a reconceptualization of assurance itself—not merely as a process of validation, but as a generative and adaptive sociotechnical practice.

To articulate this shift, we introduce the *Polycentric Generative-Assurance Theory* (PGAT). PGAT explains how trust in AI-generated code emerges through the interplay of five core assurance responsibilities, each addressing a distinct cluster of challenges. These responsibilities surfaced from recurring patterns of accountability and intervention in our data, not as formal roles, but as situated practices that developers and testers take on to safeguard software quality. They reflect a mode of responsible GenAI use, grounded in epistemic awareness and adaptive engagement rather than predefined job functions. In this view, responsible use of GenAI in software development entails actively enacting these assurance responsibilities as part of everyday workflows. Rather than relying solely on predefined

governance structures, these practices form a decentralized, dynamic model of assurance governance—one that balances the benefits of automation with the demands of human oversight, ethical judgment, and continuous validation:

1. Epistemic Mapping Responsibility. Ensures that zones of uncertainty, ambiguity, or hidden assumptions within the GenAI-generated codebase are identified and made transparent, enabling informed review and traceable decision-making.

2. Adversarial Socio-Technical Analysis Responsibility. Anticipates and inspects how GenAI-generated code could be misused, exploited, or destabilized through unintended interactions between human users, systems, and AI, thus safeguarding against socio-technical vulnerabilities.

3. Meta-Validation Responsibility. Critically evaluates the credibility and completeness of validation processes, including test frameworks and quality metrics, to ensure that what is “verified” is also meaningfully reliable.

4. Computational Ethics Responsibility. Interprets the ethical implications embedded in the code’s logic, outcomes, and use contexts, ensuring that value alignment, fairness, and accountability are actively considered in deployment.

5. Evolutionary Governance Responsibility. Iteratively designs and adapts assurance protocols in response to changing technologies, stakeholder expectations, and operating environments, sustaining governance agility over time.

These assurance responsibilities function polycentrically; they are autonomous yet interdependent, creating a dynamic network of checks, balances, and adaptive learning. Trustworthiness emerges not from any single responsibility but from the interwoven patterns of interaction among them, forming a resilient assurance architecture.

This approach to software development assurance is grounded in pragmatic constructivism, which views knowledge not as static truth but as an enacted practice, and complex adaptive governance, which treats rules as living, reflexive feedback loops. Within this framing, assurance is not about retroactive verification but about proactive orchestration of trust, in which each role offsets a distinct cluster of assurance challenges related to uncertainty, adversarial risk, ethical opacity, or systemic fragility.

Through this lens, GenAI-generated code is reimagined, not as an opaque technical output but as a continuously trustworthy sociotechnical asset. This reframing requires organizations to reassess how assurance teams are structured and how governance evolves as codebases become increasingly AI-

authored and context-sensitive. Accordingly, we put forward six propositions.

Epistemic Mapping Effect. When developers assumed responsibility for making the generative process intelligible—by identifying model assumptions, surfacing implicit logic, and documenting AI decision pathways—they significantly improved team understanding and debugging speed. These actions reflect an epistemic mapping responsibility, which fills gaps left by conventional documentation and static testing tools. Participants frequently described these efforts as “charting the unknowns” or “translating AI logic into something graspable.”

Proposition 1. Higher levels of epistemic mapping reduce black-box comprehension gaps and accelerate defect resolution by clarifying opaque regions of GenAI-generated code.

Socio-Technical Preemption Effect. Anticipating how GenAI-generated code might be misused (legally, socially, or operationally) was not simply a compliance activity but a proactive form of assurance. Practitioners who engaged in such foresight were better equipped to prevent accountability crises and licensing conflicts. These behaviors illustrate a socio-technical analysis responsibility, often emerging in response to the absence of clear regulatory precedents for GenAI outputs.

Proposition 2. Proactive socio-technical analysis and anticipation reduce post-release legal, licensing, and accountability issues, thereby enhancing organizational legitimacy.

Reflexive Validation Spiral. Conventional testing pipelines were not keeping pace with the volume and variability of AI-generated outputs. Those who took on the meta-validation responsibility—questioning not just what is tested, but how and why—helped their teams identify cascading tool failures and blind spots. This reflexive stance was crucial in reducing false positives and defect escapes as generative pipelines scaled.

Proposition 3. Greater depth in meta-validation reduces defect-escape rates amid increasing volumes of AI-generated tests, enabling more scalable and trustworthy testing.

Ethical Resonance Effect. Trust in AI-generated code was often shaped less by technical performance and more by how clearly values were embedded in the system’s design and output. Team members who assumed the computational ethics responsibility translated ethical principles into concrete system behaviors—e.g., guardrails against bias or

transparency in recommendation logic. These actions helped buffer teams against stakeholder skepticism and ethical scrutiny.

Proposition 4. *Higher ethical embedding elevates stakeholder trust and reduces security vulnerabilities, independent of codebase size or complexity.*

Governance Plasticity Advantage. Assurance protocols quickly became outdated unless someone actively re-evaluated and adapted them. This evolutionary governance responsibility emerged as critical in teams that sustained performance under rapid iteration. These individuals viewed assurance not as a one-time design artifact but as a continuously evolving system of rules, feedback, and response.

Proposition 5. *Improved governance adaptivity shortens recovery time from emergent vulnerabilities and enhances the maintainability of AI-generated code.*

Table 2.2. Key Definitions

CONCEPT	DEFINITION
Epistemic Mapping	The process of surfacing and clarifying unknowns, assumptions, and knowledge gaps within AI-generated code to enhance interpretability and traceability. Enacted through the <i>Epistemic Mapping Responsibility</i> .
Socio-Technical Analysis	The anticipatory examination of legal, social, and contextual risks arising from the deployment or misuse of GenAI-generated code. Enacted through the <i>Adversarial Socio-Technical Analysis Responsibility</i> .
Meta-Validation	The critical auditing of validation tools, processes, and assumptions used to evaluate AI-generated outputs, ensuring trust in testing itself. Enacted through the <i>Meta-Validation Responsibility</i> .
Ethical Embedding	The deliberate integration of ethical principles, such as fairness, transparency, and accountability, into the design and logic of generated code. Enacted through the <i>Computational Ethics Responsibility</i> .
Governance Adaptivity	The continuous refinement of assurance protocols and decision rules in response to changing use cases, risks, and stakeholder expectations. Enacted through the <i>Evolutionary Governance Responsibility</i> .
Ecological Complementarity	The synergistic effect produced when all assurance responsibilities interact, where each construct's value is amplified by the presence of others. Emerges through <i>cross-responsibility interdependence</i> .

Ecological Complementarity Principle. Finally, no single responsibility can consistently guarantee trustworthiness. The most resilient and trustworthy systems emerged when multiple responsibilities were enacted in concert. The strength of epistemic mapping reinforced the depth of validation; ethical embedding tempered governance rigidity; and socio-technical anticipation exposed new validation needs. This

ecological complementarity reflects the necessity of a polycentric structure, in which distributed responsibilities interact and co-evolve.

Proposition 6. *System trustworthiness increases most significantly when all five assurance responsibilities—mapping, anticipation, meta-validation, ethical embedding, and adaptivity—are enacted in concert, as their effects are mutually reinforcing and exhibit ecological complementarity.*

6 Theoretical Contributions

This study advances the theoretical frontier of information assurance by reconceptualizing assurance as a polycentric, generative process attuned to the unique uncertainties of AI-generated code. Prevailing models of information assurance have predominantly focused on deterministic systems, emphasizing predefined controls, static validation pipelines, and role-based accountability. Our findings challenge this paradigm by surfacing five emergent assurance responsibilities—epistemic mapping, socio-technical analysis, meta-validation, computational ethics, and evolutionary governance—that operate not as linear checkpoints but as dynamic, interdependent functions within a sociotechnical ecosystem. In doing so, we contribute to the literature in three significant ways.

First, we introduce a novel constructive vocabulary for information assurance in generative coding environments and their governance. Rather than relying on traditional roles (e.g., tester, auditor), our study identifies responsibility constructs that better reflect the fluid, overlapping, and context-sensitive nature of assurance work. These constructs are not only empirically grounded but conceptually distinct, offering new levers for both analytical and design-based inquiry.

Second, we articulate relational logic among these responsibilities. PGAT reveals how each responsibility addresses a specific cluster of assurance challenges and how their interactions create systemic resilience. This polycentric configuration marks a departure from conventional assurance models, which are predicated on hierarchical control or singular oversight functions, thereby offering a more realistic and robust foundation for research.

Third, we expand the conceptual boundaries of information assurance itself. By incorporating principles from pragmatic constructivism and complex adaptive governance, we reposition assurance not as a downstream activity of error detection, but as a generative practice of trust orchestration. This shift invites future research to explore assurance as an ongoing, co-evolutionary process shaped by human judgment, organizational learning, and ethical

negotiation. It also opens new lines of inquiry into how assurance responsibilities can be instantiated, shared, and scaled across hybrid human–AI development teams.

Together, these contributions reframe assurance and responsible governance in the age of generative code—from a static checklist of safeguards to a living architecture of distributed responsibility. In doing so, our work lays a theoretical foundation for scholars to investigate how assurance systems can adapt, evolve, and remain trustworthy in the face of the emergent and ethically complex nature of GenAI systems.

7 Practical Implications

Our findings suggest that organizations must fundamentally rethink their information assurance practices in the era of GenAI. Traditional role-based approaches, which isolate testing, validation, and compliance into discrete functions, are no longer sufficient. Instead, organizations should adopt a responsibility-centered model, where assurance functions—such as epistemic mapping, socio-technical analysis, and ethical embedding—are treated as shared, adaptive responsibilities that may shift across individuals and teams over time. This shift requires reconfiguring team structures to support cross-functional collaboration and enable the fluid enactment of assurance practices, where individuals can take on multiple responsibilities depending on the context. Our findings show that assurance starts at code generation, positioning developers on the front line of responsibility. As key interfaces with GenAI tools, developers shape the trustworthiness of code and should be supported through training that builds not only technical skills but also reflexive judgment, ethical reasoning, and adaptability in governance.

8 Limitations and Future Research

This study is limited by its reliance on self-reported and public Reddit data, which may overlook tacit practices and organizationally sensitive routines. Findings are situated in agile, product-oriented workflows, restricting their transferability to safety-critical, financial, or open-source settings. The exploratory, phenomenon-driven design, while well-suited to surface emergent responsibilities, limits causal inference and predictive generalizability. The use of semantic clustering and LLM-assisted coding introduces interpretive bias, even with iterative validation and human oversight. Finally, the rapid pace of GenAI tool development and regulatory change risks rendering static assurance models quickly obsolete.

Accordingly, future research can advance along four interrelated domains. First, methodological depth: ethnographic, proprietary, and longitudinal studies are needed to uncover hidden practices and trace how assurance responsibilities evolve over time. Second, cross-context generalization: comparative investigations across safety-critical, highly regulated, and open-source settings can help establish the boundary conditions and contextual variations of PGAT. Third, temporal adaptivity: living-lab approaches are well-suited to capture shifting responsibilities and governance responses as GenAI tools and policies continue to evolve. Finally, organizational mechanisms: future work should examine how responsibilities are distributed, supported, and institutionalized through training, workflow design, and policy—leveraging experimental, simulation, and design science methods to explore effective implementation at scale.

Acknowledgement. This work was partially supported by the U.S. National Science Foundation under Grant No. 2219773. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

9 References

- Banh, L., Holldack, F., & Strobel, G. (2025). Copiloting the future: How generative AI transforms software engineering. *Information and Software Technology, 183*, 107751.
- Bendig, D., & Bräunche, A. (2024). The role of artificial intelligence algorithms in information systems research: A conceptual overview and avenues for research. *Management Review Quarterly*.
- Chen, M., Chua, A. Y. K., & An, L. (2025). Archetypes of influential users in social question-answering sites. *Internet Research, 35*(1), 419–447.
- Donvir, A., Panyam, S., Paliwal, G., & Gujar, P. (2024, October). The role of generative AI tools in application development: A comprehensive review of current technologies and practices. *2024 International Conference on Engineering Management of Communication and Technology* (pp. 1–9). IEEE.
- Drori, I., & Te'eni, D. (2024). Human-in-the-loop AI reviewing: Feasibility, opportunities, and risks. *Journal of the Association for Information Systems, 25*(1), 98–109.
- Dwivedi, Y. K., Hughes, L., Kshetri, N., Slade, E. L., Jeyaraj, A., Kar, A. K., Wright, R. (2023). “So what if ChatGPT wrote it?” Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. *International Journal of Information Management, 71*, 102642.

- Ebrahim, S., & Tanner, M. (2025). A systematic literature review on the use of ChatGPT to support productivity of software developers. *AMCIS 2025 Proceedings*, 17.
- Gregory, R. W., & Henfridsson, O. (2021). Bridging art and science: Phenomenon-driven theorizing. *Journal of the Association for Information Systems*, 22(6), 1509-1523.
- Harding, B. (2025). *AI Copilot code quality: 2025 look back at 12 months of data*. GitClear.
- IEEE. (2008). *Standard for software reviews and audits*. Institute of Electrical and Electronics Engineers.
- Jackson, V., Vasilescu, B., Russo, D., Ralph, P., Prikladnicki, R., Izadi, M., ... van der Hoek, A. (2025). The impact of generative AI on creativity in software development: A research agenda. *ACM Transactions on Software Engineering and Methodology*, 34(5), 1–28.
- Jin, X., & Lin, Z. (2024). SimLLM: Calculating semantic similarity in code summaries using a large language model-based approach. *Proceedings of the ACM on Software Engineering*, 1, 62.
- Kumar, S., Datta, S., Singh, V., Datta, D., Singh, S. K., & Sharma, R. (2024). Applications, challenges, and future directions of human-in-the-loop learning. *IEEE Access*, 12, 75735–75760.
- Mayer, A. S., Kostis, A., Strich, F., & Holmström, J. (2025). Shifting dynamics: How generative AI as a boundary resource reshapes digital platform governance. *Journal of Management Information Systems*, 42(3), 767-798.
- Mucha, T., Ma, S., & Abhari, K. (2023). Riding a bicycle while building its wheels: the process of machine learning-based capability development and IT-business alignment practices. *Internet Research*, 33(7), 168-205.
- Negi, K., Kota, M., Ramesh, B., & Cao, L. (2025). The paradoxical impact of generative AI on agile software development outcomes: A process view. *AMCIS 2025 Proceedings*, 9.
- Paradis, E., Grey, K., Madison, Q., Nam, D., Macvean, A., Meimand, V., & Chandra, S. (2024). How much does AI impact development speed? An enterprise-based randomized controlled trial. *arXiv:2410.12944*.
- Parteka, A., & Kordalska, A. (2023). Artificial intelligence and productivity: global evidence from AI patent and bibliometric data. *Technovation*, 125, 102764.
- Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., & Santos, E. A. (2024). “It’s weird that it knows what I want”: Usability and interactions with Copilot for novice programmers. *ACM Transactions on Computer-Human Interaction*, 31(1), Article 4.
- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Rivera, A., Abhari, K., & Xiao, B. (2025). Responsible AI Design: The Authenticity-Control-Transparency Theory. *Journal of the Association for Information Systems*, 26(5), 1337-1389.
- Sadowski, C., Söderberg, E., Church, L., Sipko, M., & Bacchelli, A. (2018). Modern code review: A case study at Google. *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice* (pp. 181–190). Association for Computing Machinery.
- Sergeyuk, A., Golubev, Y., Bryksin, T., & Ahmed, I. (2025). Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward. *Information and Software Technology*, 178, 107610.
- Software Assurance Forum for Excellence in Code. (2019). *Principles for software assurance assessment*.
- Söllner, M., Arnold, T., Benlian, A., Berente, N., Berger, B., Buxmann, P., Grover, V., Janson, A., Kundisch, D., Maedche, A., Müller, B., Susarla, A., & Venkatesh, V. (2025). ChatGPT and beyond: Exploring the responsible use of generative AI in the workplace. *Business & Information Systems Engineering*, 67(3), 289–303.
- Stohr, A., Ollig, P., Keller, R., & Rieger, A. (2024). Generative mechanisms of AI implementation: A critical realist perspective on predictive maintenance. *Information and Organization*, 34(2), 100503.
- Subramanya, S., & Paul, A. (2024). Promises and pitfalls: Evaluating generative AI tools in software engineering. *Proceedings of the Australasian Conference on Information Systems*.
- Sunyaev, A., Benlian, A., Pfeiffer, J., et al. (2025). High-risk artificial intelligence. *Business & Information Systems Engineering*.
- Tabarsi, B., Reichert, H., Limke, A., Kuttal, S., & Barnes, T. (2025). LLMs' reshaping of people, processes, products, and society in software development: A comprehensive exploration with early adopters. *arXiv preprint arXiv:2503.05012*.
- Vasist, P. N., Krishnan, S., & Agnihotri, P. (2025). The unfolding of geopolitical tensions on social networks: A social network analysis of Twitter and Reddit conversations. *Internet Research*. Advance online.
- Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., & Wei, F. (2023). Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.
- Wang, A. Y., Wang, D., Drozdal, J., Muller, M., Park, S., Weisz, J. D., ... Dugan, C. (2022). Documentation matters: Human-centered AI system to assist data science code documentation in computational notebooks. *ACM Transactions on Computer-Human Interaction*, 29(2), 1–33.
- Wang, W., Gao, G. (Gordon), & Agarwal, R. (2023). Friend or foe? Teaming between artificial intelligence and workers with variation in experience. *Management Science*, 70(9), 5753–5775.
- Weber, T., Brandmaier, M., Schmidt, A., & Mayer, S. (2024). Significant productivity gains through programming with large language models. *Proceedings of the ACM on Human-Computer Interaction*, 1–29.
- Winter, M., & Wright, A. (2020). How to do a code review. In T. O’Sullivan & H. Wright (Eds.), *Software engineering at Google: Lessons learned from programming over time* (pp. 199–230). O’Reilly Media.