Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

The AI Assurance Gap (AAG): A Framework for Measuring Security Review Deficits in AI-Assisted Software Development within Regulated Financial Institutions

Dr. Sadham Hussain Razzak

sadham@hostershouse.com

Kennedy University

Article

Keywords: generative AI, AI-assisted software development, cybersecurity assurance, secure software development, AI governance, financial institutions, operational resilience, security review, DevSecOps, risk measurement

Posted Date: November 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11377221/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Abstract

Generative artificial intelligence (GenAI) is increasingly used to generate, transform, test, document, and configure software, increasing development throughput while changing the workload presented to security assurance functions. In regulated financial institutions, the critical governance question is whether security review, validation, and accountability can scale with AI-assisted change. This paper introduces the AI Assurance Gap (AAG), a conceptual framework for measuring the deficit between risk-weighted security assurance demand and effective assurance capacity. The framework defines an AAG Ratio, AAG Deficit, AAG Coverage, Assurance Backlog, and Assurance Latency. Assurance demand is normalized using change volume, technical complexity, business criticality, and exposure, while capacity is based on observed control throughput rather than nominal staffing. A synthetic fintech scenario illustrates how persistent demand above capacity can accumulate assurance debt even when automated testing remains active. The framework is mapped to contemporary AI, cybersecurity, software-development, and financial-sector governance instruments. AAG is intentionally positioned as a domain-specific security-assurance operationalization of the broader generation-versus-verification problem. Empirical validation with longitudinal organizational data is proposed as the next research stage.

1. Introduction

Generative AI has changed the economics and mechanics of software development. Modern coding assistants can propose functions, refactor modules, create tests, explain code, generate infrastructure-as-code, and support developers across a continuous delivery workflow. Empirical work has repeatedly demonstrated meaningful productivity effects. In a controlled experiment, developers using GitHub Copilot completed an HTTP-server task 55.8% faster than the control group [1]. More recently, three field experiments covering 4,867 developers reported a combined 26.08% increase in completed tasks for developers given access to an AI coding assistant [2]. A quasi-experiment involving Ant Group's CodeFuse similarly reported an increase of more than 50% in code output [3]. These findings matter for cybersecurity because software assurance is performed by a different production function: application security engineers, architects, testers, auditors, risk teams, developers acting as reviewers, and automated security controls.

The key policy question therefore shifts from “Is AI-generated code secure?” to a second-order governance question: “Can the institution assure the security of AI-assisted changes at the rate and depth required by its risk appetite?” Existing research provides evidence that AI-generated or AI-assisted code may exhibit security weaknesses, and that developers may be overconfident in the security of generated code [4–6]. Pearce et al. found that approximately 40% of programs produced in their high-risk scenario set were vulnerable [4]. Fu et al. later analyzed 733 AI-generated snippets from real GitHub projects and reported security weaknesses across 43 CWE categories, including 29.5% of the Python snippets and 24.2% of JavaScript snippets in their study [5]. A 2025 ACM study of 105 developers found increased development speed alongside concerns about security, overreliance, and additional time spent

reviewing AI-generated code [6]. These studies establish a security quality problem, but quality defects alone do not explain an institutional assurance deficit.

The distinction is important for regulated financial institutions. A bank, broker, custodian, payment firm, or fintech may already require peer review, security testing, vulnerability management, penetration testing, change control, segregation of duties, audit evidence, and incident reporting. If GenAI increases the number of software changes entering these control points without a proportional increase in assurance capacity, the institution can experience a widening difference between changes produced and changes adequately assured. That condition can exist even when each individual pipeline contains nominally correct controls. A queue may form behind human review, architecture approval, exception handling, or deeper security analysis; automated scanners may process every change yet fail to provide the contextual assurance expected for high-impact logic; and teams may approve larger changes with less review depth because the generated output is fluent and difficult to interrogate.

This paper names that condition the AI Assurance Gap (AAG). The construct does not claim that the general idea that generation can exceed verification was previously absent from the literature. A June 2026 SSRN paper, for example, proposed a broader "Verification Rate Gap" in which development risk emerges when AI-generated outputs are produced faster than they can be reliably verified [7]. Recent practitioner research likewise describes code-review bottlenecks and trust-calibration challenges for large AI-generated multi-file changes [8,9]. The contribution of this paper is narrower and more operational: it translates the broad verification imbalance into a security-assurance metric designed for regulated financial software delivery, incorporates risk-weighted assurance demand and measured control capacity, and connects the metric to governance and resilience requirements.

The remainder of the paper is organized as follows. Section 2 reviews related research and standards. Section 3 defines the research problem, questions, and hypotheses. Section 4 establishes the theoretical framing. Section 5 presents the AAG framework. Section 6 develops the quantitative measurement model. Section 7 illustrates application in a regulated financial institution using synthetic data. Section 8 maps the framework to major governance and regulatory instruments. Section 9 discusses implementation, benefits, and limitations. Section 10 proposes an empirical validation protocol. Section 11 concludes.

2. Background and Literature Review

2.1 AI-Assisted Software Development and Productivity

The adoption of AI coding assistants is associated with increased developer throughput, shorter task completion times, and changes in how developers interact with software construction. Peng et al. reported a 55.8% task-completion improvement in a controlled experiment [1]. Cui et al. found a 26.08% increase in completed tasks across three randomized field experiments involving 4,867 developers [2]. Barke et al. observed two recurring modes of interaction with Copilot: "acceleration," where developers

already know what they want to do, and “exploration,” where the assistant is used to discover possible approaches [10]. This distinction is relevant to security assurance because exploratory use can expand not only the speed of coding but also the solution space that later must be understood and assessed.

Productivity evidence should not be interpreted as uniformly positive or uniformly negative. Copilot and related tools can reduce routine effort and enable more code to be produced, but they also alter the size, provenance, and cognitive characteristics of software changes. In a banking environment, for example, a developer may use AI to generate an API handler, tests, an infrastructure module, and a deployment workflow during one development session. A conventional PR-based control model may treat the output as one change request even though the assurance burden is distributed across application logic, cloud configuration, identity permissions, and deployment behavior. This is precisely the kind of mismatch that motivates an assurance-demand model rather than a simple “lines of code reviewed” metric.

2.2 Security Quality and Vulnerability Evidence

Research on AI-generated code has repeatedly found security weaknesses, although results vary substantially by model, language, task, prompting strategy, and evaluation method. Pearce et al. systematically tested Copilot using scenarios connected to high-risk CWE categories and found approximately 40% of generated programs vulnerable in their experimental set [4]. Perry et al. conducted a user study and found that participants with an AI assistant wrote significantly less secure code and were more likely to believe their code was secure [11]. Fu et al. extended the evidence into real GitHub projects and found weaknesses in AI-generated snippets across dozens of CWE categories [5]. Their work also showed that AI-assisted repair could fix a substantial fraction of identified issues, illustrating that GenAI may act as both a source of defects and a tool for remediation.

More recent work continues to emphasize validation after generation. Pandey and Kumar evaluated several open-source generative models on secure Python-code tasks and highlighted recurring security weaknesses and the importance of post-generation validation [12]. NIST has also expanded its secure-development guidance by publishing an SSDF Community Profile for generative AI and dual-use foundation models, emphasizing secure practices across development lifecycles [13]. OWASP guidance similarly recommends that AI-generated code have a human owner and undergo review and security testing, and the AI Security Verification Standard includes requirements for qualified human review, automated testing, and elevated scrutiny for security-critical code [14,15]. The convergence of these sources is notable: AI-assisted generation does not remove assurance work; in many contexts it makes assurance more important.

2.3 Human Oversight, Review Load, and Trust Calibration

Human review is not a binary event. A reviewer can approve a change after a shallow inspection or conduct a deep analysis of business logic, authorization boundaries, error handling, dependency behavior, and failure modes. AI-generated changes may also be less interpretable through social cues than work authored directly by a developer. Recent research from JetBrains and Lund University describes a review problem framed around trust calibration: reviewers must allocate attention

proportionally to risk when the “author” is an AI system that cannot provide human-style contextual explanations [8]. OpenAI researchers have separately argued that the volume of AI-produced code can exceed the limits of thorough human oversight and that scalable automated verification should complement human controls [9]. The broader 2026 Verification Rate Gap proposal similarly models development risk as a function of the difference between generation rate and verification rate [7].

The AAG framework builds on these observations but introduces two distinctions. First, “verification” is narrowed to security assurance rather than all forms of correctness or evaluation. Second, capacity is measured in weighted assurance units, allowing an institution to account for the fact that reviewing a customer-facing authentication module is not equivalent to reviewing a low-risk formatting change. This makes the construct more compatible with risk-based financial-sector governance.

2.4 Financial-Sector Governance and Operational Resilience

Financial institutions operate under layered expectations for technology risk, cyber risk, resilience, third-party risk, and governance. The Financial Stability Board's June 2026 consultation report identified twelve sound practices for responsible AI adoption by financial institutions, including organization-wide governance, AI lifecycle risk management, and AI-related cybersecurity, ICT, and third-party risks [16]. The Bank of England has also described AI use in finance as already including activities such as code generation, while emphasizing that AI can alter operational and financial risk profiles [17]. These developments strengthen the case for an assurance measurement that can sit inside an existing enterprise-risk process rather than beside it.

In the European Union, DORA applies to a broad set of financial entities and harmonizes digital operational resilience requirements, including ICT risk management and incident reporting [18]. The EU AI Act establishes a risk-based legal framework for AI systems and, as of 2 August 2026, the regulation has reached its principal general application date, subject to the staged provisions specified in Article 113 [19]. ISO/IEC 27001 provides the management-system foundation for information security risk treatment [20], while ISO/IEC 42001 provides requirements for an AI management system [21], and ISO/IEC 23894 provides guidance for AI risk management [22]. NIST AI RMF 1.0 and its Generative AI Profile provide a voluntary structure for identifying and managing AI-related risks [23,24]. NIST SSDF provides a complementary software-security framework, with the generative-AI profile in SP 800-218A extending secure-development guidance to AI-specific lifecycle considerations [13,25].

The ADGM/FSRA context is especially relevant to this paper because it illustrates how technology and cyber risk are being incorporated into financial-sector supervision. ADGM published Information Technology Risk Management Guidance in November 2024, and the FSRA issued a Cyber Risk Management Framework in July 2025 with compliance required from 31 January 2026 [26,27]. The FSRA also describes IT risk management as part of overall business risk and expects regulated entities to use risk-based approaches proportionate to vulnerabilities, complexity, and technology reliance [28]. The

AAG framework is therefore designed to complement this risk-based logic rather than introduce a parallel compliance system.

2.5 Gap in Existing Literature

Table 1
Positioning of the AAG framework relative to adjacent research.

Research stream	Primary question	Typical unit of analysis	Observed limitation for regulated finance
AI productivity	Does AI increase developer throughput?	Task, PR, LOC, completion time	Does not quantify assurance capacity
AI code security	Is AI-generated code vulnerable?	Snippet, program, CWE finding	Does not model organizational review queues
Human-AI review	How should people review generated code?	Reviewer behavior, trust, UX	Limited operational metric for assurance demand
AI governance	How should AI risks be managed?	Policy, lifecycle, control	Often lacks software-assurance flow metric
Verification-rate gap	What happens when generation exceeds verification?	Generation/verification rate	General construct; not tailored to financial security assurance
AAG proposal	Can security assurance demand and capacity be measured together?	Weighted assurance unit and capacity flow	Purpose-built for regulated software security

The literature therefore contains the ingredients for AAG but not the exact operational construct proposed here. The novelty claim is intentionally scoped: AAG is presented as a domain-specific framework and measurement model for security assurance in AI-assisted software development in regulated financial institutions. The paper does not claim to be the first work to observe the generation-versus-verification imbalance.

3. Materials and Methods

3.1 Research Problem

The research problem is an asymmetry between the rate at which software artifacts can be generated or modified with AI assistance and the rate at which those artifacts can receive adequate security assurance. The asymmetry is operationally important because security assurance is constrained by scarce resources and context-dependent work: experienced reviewers, threat-modeling expertise, security testing infrastructure, architecture knowledge, compliance interpretation, and time to investigate findings. Increasing code production without increasing these inputs can create a queue of changes that are not equally assured.

Let a development period t contain a set of AI-assisted change artifacts A_t. Each artifact a_i contributes a weighted amount of assurance demand d_i. Let the institution's realized assurance capacity during that period be C_t. A gap exists whenever total demand exceeds the effective capacity available to process it. The core proposition is that persistent positive gaps create assurance debt, analogous to technical or vulnerability debt, except that the debt is specifically the accumulation of risk-bearing change that has not received the required level of assurance.

3.2 Research Questions

- RQ1. How can security assurance demand generated by AI-assisted software changes be normalized into a measurable unit that is independent of programming language?
- RQ2. How can an organization estimate effective security assurance capacity using observed review and validation throughput rather than staffing counts alone?
- RQ3. How does the ratio between assurance demand and capacity relate to backlog, assurance latency, and residual security risk?
- RQ4. What governance actions should a regulated financial institution take when the AAG exceeds defined risk thresholds?

3.3 Conceptual Research Method

This first paper is conceptual and follows a design-oriented framework-development approach. The method has four stages. Stage 1 synthesizes empirical research on GenAI productivity, code-security weaknesses, and human review. Stage 2 extracts control concepts from contemporary standards and regulatory guidance. Stage 3 develops an operational model consisting of variables, equations, scoring rules, and governance actions. Stage 4 demonstrates the model using synthetic data designed to reveal how a gap develops over time. The synthetic example is not presented as empirical evidence and should not be interpreted as a validation result.

The next research stage should be empirical: collecting longitudinal, anonymized data from software repositories and security tooling to test whether AAG measures are associated with review latency, escaped vulnerabilities, audit findings, change-failure rates, or other security outcomes. This separation between framework development and validation is deliberate. It avoids fabricating organizational results merely to make the conceptual paper appear more complete.

3.4 Hypotheses for Future Empirical Testing

- H1. Higher AI-assisted development throughput is positively associated with AAG Ratio when assurance capacity is held approximately constant.
- H2. Higher AAG Ratio is positively associated with assurance backlog and assurance latency.
- H3. Higher sustained AAG Deficit is associated with higher rates of escaped security defects or material control exceptions.

- H4. Risk-weighted assurance capacity, automated security testing, and explicit human approval gates reduce the relationship between AI-assisted throughput and AAG Deficit.
- H5. Institutions with formal AI governance and traceability practices experience lower AAG variability than institutions with ad hoc AI use.

3.5 Theoretical Framing

3.5.1 Assurance as a Flow Problem

Software delivery can be represented as a flow system. AI-assisted generation increases the arrival rate of change artifacts into review and validation stages. Assurance activities form one or more service stations: peer review, application-security review, architecture assessment, static and dynamic testing, dependency analysis, infrastructure scanning, penetration testing, compliance checks, and exception handling. Each station has a finite processing capacity and a different ability to detect or mitigate classes of risk.

The AAG model therefore treats assurance as a flow rather than a checklist. A checklist can show that “code review is required” while hiding whether review demand has expanded from 100 weighted units to 800 weighted units per month. The existence of a control does not prove that its service capacity is sufficient under a new production regime. This distinction is central to the paper.

3.5.2 Assurance Demand Versus Assurance Capacity

Assurance demand is the weighted quantity of work that must be performed to provide the level of security confidence required by the organization's risk appetite. Capacity is the volume of equivalent assurance work the organization can complete with its available people, tools, review windows, testing infrastructure, and specialized expertise. Importantly, capacity should be estimated from observed throughput. A team of three engineers may nominally have 480 hours per month, but if only 180 effective hours are available for review after meetings, incident response, architecture work, leave, and rework, the practical assurance capacity is much lower.

The model also distinguishes automated assurance from human assurance. Automation should increase capacity when it detects or validates relevant control conditions, but raw scanner execution must not be equated with assurance completion. A SAST scan that runs in two minutes is not equivalent to a contextual review of authentication logic. The framework therefore uses an “effective assurance throughput” concept: automated checks contribute capacity only to the extent that the organization has calibrated their coverage, false-positive behavior, and decision criteria against its assurance objectives.

3.5.3 Assurance Debt

Assurance debt is the accumulated weighted quantity of change that has not yet received the required assurance level. It differs from vulnerability debt because an artifact can be debt-bearing even when no known vulnerability has been identified. An unreviewed cryptographic implementation, an unvalidated

IAM policy, or an AI-generated deployment workflow may be perfectly safe—or unsafe—but the organization cannot establish that fact with the intended level of assurance until the control is performed.

This distinction creates a useful management signal. A firm may report a declining count of open vulnerabilities while its assurance debt rises because more changes are reaching production with limited contextual review. AAG is intended to reveal that hidden condition.

4. Results

4.1 Construct Definition

The AI Assurance Gap (AAG) is defined as the measurable deficit between risk-weighted security assurance demand generated by AI-assisted software changes and the organization's effective security assurance capacity during a defined reporting period. The framework treats AI attribution as a provenance dimension rather than assuming that every AI-generated artifact is inherently more dangerous. The risk score is driven by complexity, criticality, and exposure; AI involvement changes the governance need for traceability and review but does not automatically inflate risk.

4.2 Framework Architecture

Table 2
AAG conceptual architecture.

Layer	Purpose	Primary evidence
AI Generation	Identify where AI contributes to a change	AI tool/model, prompt context where retained, provenance flag
Development	Normalize the change into assurance units	PR, commit, files/modules, IaC, policy/configuration changes
Assurance	Apply the required security controls	Human review, SAST, DAST, SCA, IaC scan, threat model, pentest
Governance	Evaluate residual assurance deficit	AAG Ratio, AAG Deficit, Assurance Backlog, latency, exceptions
Regulatory/Resilience	Escalate according to risk and obligations	Risk appetite, materiality, incident obligations, audit evidence

4.3 Assurance Unit Normalization

Counting lines of code is attractive because it is easy to obtain, but LOC is a weak proxy for assurance effort across modern software systems. A 20-line authorization rule may carry greater risk than 500 lines of test scaffolding; a small Terraform change may create broad privileges; a one-line cryptographic parameter change may have systemic effects. AAG therefore uses an assurance-unit approach.

For each change artifact i, define a base unit U_i representing the amount of change. U_i may be derived from a calibrated combination of pull-request size, touched modules, infrastructure objects, identity objects, and other organization-specific indicators. Define three ordinal risk dimensions: complexity C_i, criticality K_i, and exposure E_i, each normalized to [1,5]. A simple initial weighting is:

$W_i = 0.30(C_i) + 0.40(K_i) + 0.30(E_i)$ (1)

The weighting emphasizes business criticality while retaining technical complexity and exposure. The 0.30/0.40/0.30 coefficients are proposed starting values, not empirically validated constants. They should be calibrated using historical security incidents and review effort. The assurance demand for artifact i is then:

$D_i = U_i \times W_i$ (2)

Total assurance demand for period t is:

$AD_t = \Sigma D_i$ (3)

4.4 Effective Assurance Capacity

Let each assurance control family j have an observed effective throughput rate R_j measured as weighted assurance units completed per unit of available service time. Let T_j represent usable service time during the reporting period. Effective assurance capacity is:

$AC_t = \Sigma (R_j \times T_j)$ (4)

The model can include human review, automated testing, architecture review, penetration testing, and compliance validation as separate service stations. A control contributes to capacity only when it has defined evidence and acceptance criteria. For example, a SAST engine may provide high-throughput detection capacity for a subset of vulnerabilities, while a security architect provides lower-volume capacity for design-level risk. The two capacities are not interchangeable without calibration.

4.5 Core AAG Measures

$AAG\ Ratio_t = AD_t / AC_t$ (5)

$AAG\ Deficit_t = \max(0, 1 - AC_t / AD_t) \times 100\%$ (6)

$AAG\ Coverage_t = \min(1, AC_t / AD_t) \times 100\%$ (7)

The AAG Ratio is the primary flow-pressure metric. A value of 1.00 means modeled demand and effective capacity are balanced for the period. Values above 1.00 indicate demand exceeds capacity. The AAG Deficit translates the ratio into an intuitive percentage of demand that cannot be covered by the modeled capacity during the period. AAG Coverage expresses the same state from a positive perspective.

4.6 Assurance Backlog and Assurance Latency

Flow measures alone can hide historical debt. Therefore the framework requires two stock measures. Assurance Backlog (AB) is the weighted amount of unassured change carried forward at the end of the period:

$AB_t = \max(0, AB_{t-1} + AD_t - AC_t)$ (8)

Assurance Latency (AL) is the time between a change becoming ready for the relevant assurance gate and completion of the required assurance decision. AL can be reported as a median, 75th percentile, and 95th percentile. High AAG combined with rising AL is a strong signal that the assurance system is becoming a bottleneck. High AAG with stable AL may mean the organization is increasing capacity quickly or is shifting risk to automated controls; such cases still require effectiveness analysis.

4.7 Provisional Risk Bands

Table 3
Proposed calibration bands. These are hypothesis-generating thresholds, not regulatory limits.

AAG Ratio	Provisional state	Interpretation	Suggested governance action
< 1.00	Balanced	Capacity at or above modeled demand	Maintain monitoring and calibration
1.00–1.25	Watch	Emerging pressure; limited deficit	Review near-term backlog and capacity
> 1.25–1.50	Moderate	Material assurance shortfall	Increase targeted capacity; prioritize critical changes
> 1.50–2.00	High	Assurance demand substantially exceeds capacity	Escalate to technology/security risk owner; restrict high-risk change throughput
> 2.00	Critical	Capacity is less than half the modeled demand	Executive escalation; enforce gating or controlled release

The proposed bands are intentionally provisional. A research paper should not present them as objectively validated risk thresholds before empirical testing. A mature implementation would estimate thresholds from organizational loss tolerance, historical escaped-defect rates, assurance latency, and regulatory materiality.

4.8 AI Provenance and Review Depth

AI provenance is recorded as a governance dimension. Each change may be tagged as human-authored, AI-assisted, AI-drafted and human-refined, or predominantly AI-generated. Provenance can then trigger control requirements without automatically changing the base risk score. For example, a customer-authentication module touched by AI may require independent human review because the organization

wants assurance against overreliance, regardless of whether its initial risk score is low. This design prevents the common analytical error of equating “AI-generated” with “high risk” in every context.

4.9 Governance Implementation in a Regulated Financial Institution

4.9.1 Operating Model

An AAG program can be implemented without replacing the organization's SDLC or GRC platform. The minimal data model requires four records: a change record, an assurance record, a risk record, and a capacity record. The change record stores provenance and scope; the assurance record stores which controls were executed and their evidence; the risk record stores criticality and exposure; and the capacity record stores observed service throughput. These can be drawn from source-control systems, CI/CD platforms, SAST/SCA/IaC tools, ticketing systems, GRC repositories, and security-testing schedules.

The framework should operate on a risk-based cadence. Low-risk changes can be aggregated into daily or weekly measurement windows. Critical services should be measured more frequently and may require per-change gates. A regulated institution should not use AAG as a reason to slow every software team equally. Its purpose is to distinguish where additional assurance is required because demand, exposure, or criticality has become disproportionate.

4.9.2 Minimum Control Pattern for AI-Assisted Changes

- Record AI provenance for material changes, including the AI tool/model where the organization can reliably capture it.
- Maintain independent human accountability for security and maintainability of accepted AI-assisted changes.
- Run automated security checks on every applicable change, including SAST, software composition analysis, secret scanning, infrastructure-as-code scanning, and relevant dynamic testing.
- Apply elevated review to security-critical paths, such as authentication, authorization, cryptography, payment logic, key management, transaction signing, and privileged infrastructure.
- Retain evidence sufficient to reconstruct who reviewed the change, which controls ran, what exceptions were granted, and what residual risk was accepted.
- Use AAG and backlog trends as inputs to release planning and risk decisions rather than treating them as post-hoc dashboards.

These principles are consistent with contemporary secure-development and AI-governance guidance. OWASP's AI secure-coding guidance recommends human ownership and explicit approval for AI-generated code [14]. NIST SSDF provides the lifecycle vocabulary for secure-development activities [25], and NIST SP 800-218A extends those ideas to AI and foundation-model development [13]. The AAG

construct adds a management layer by asking whether the institution has enough capacity to perform those controls as AI-driven change volume grows.

4.9.3 Governance Decision Rules

Table 4
Example AAG governance decision rules.

Condition	Interpretation	Management response
AAG > 1.25 for 1 period	Transient pressure	Confirm capacity assumptions; inspect backlog
AAG > 1.25 for 3 periods	Sustained deficit	Capacity plan and risk-owner escalation
AAG > 1.50 and AL p95 rising	Queue stress	Prioritize critical paths; restrict nonessential high-risk releases
AAG > 2.00	Critical shortfall	Executive approval for high-risk releases; temporary change gating
AB rising despite AAG < 1	Backlog stock remains	Clear debt; do not infer current balance means historical debt is gone
High AAG but low security findings	Possible control blind spot	Validate control effectiveness and scope; do not assume safety

4.10 Illustrative Synthetic Case Study

4.10.1 Scenario

Consider a hypothetical regulated fintech with a customer-facing web platform, mobile applications, payment APIs, cloud infrastructure, and internal administrative services. The institution has adopted AI coding assistants for routine development and is beginning to use AI agents for multi-file change generation. The example is synthetic and contains no data from a real organization.

Assume the baseline month contains 120 AI-assisted pull requests. After normalization, the average change is worth 3.0 assurance units, producing 360 weighted assurance units before complexity/criticality adjustments. The organization has two application-security reviewers, automated analysis, and a periodic architecture/pentest function. After calibrating observed throughput, the effective monthly capacity is estimated at 260 assurance units.

AAG Ratio = 360 / 260 = 1.38 (9)

AAG Deficit = (1 − 260 / 360) × 100% = 27.8% (10)

Under the provisional bands, this month is in the moderate range. The interpretation is not that 27.8% of code is insecure. It means that the modeled assurance work exceeds the effective capacity by an amount equivalent to 27.8% of demand. Some of the uncovered demand may be addressed by

automation, deferred review, risk acceptance, or changes in scope. The metric exists to force those choices into view.

4.10.2 Growth Scenario

Table 5
Synthetic three-month AAG progression.

Metric	Month 1	Month 2	Month 3
AI-assisted changes	120	180	240
Avg. base assurance units/change	3.0	3.0	3.2
Assurance demand (weighted units)	360	540	768
Effective assurance capacity	260	280	300
AAG Ratio	1.38	1.93	2.56
AAG Deficit	27.8%	48.1%	60.9%
Illustrative interpretation	Moderate	High	Critical

The synthetic scenario illustrates an important property of the framework: the organization can improve capacity from 260 to 300 units and still become materially less assured because demand is increasing faster than capacity. The dominant management problem is therefore not “how many security tools do we have?” but “is effective assurance capacity scaling with the risk-weighted change flow?”

4.10.3 Assurance Debt Interpretation

Suppose Month 1 starts with 50 weighted assurance units already in backlog. If Month 1 demand is 360 and capacity is 260, the period ends with approximately 150 units of backlog. In Month 2, backlog becomes approximately 410 units before any risk-based prioritization, and Month 3 increases it further. This is the assurance equivalent of a queue that begins to dominate the system. At some point, simply hiring more reviewers after the fact may not be enough because the backlog itself becomes a review-risk concentration.

A practical institution would therefore pair AAG with a criticality-weighted queue. High-impact changes should jump ahead of low-risk changes, while noncritical work may be deferred or subject to stricter automated gates. The objective is not to drive AAG to zero at all costs; rather, it is to prevent the organization from unknowingly operating outside its accepted assurance tolerance.

4.11 Regulatory and Standards Mapping

The AAG framework is designed to sit above existing standards as a measurement layer. It is not proposed as a replacement for any regulation or certification standard. The mapping below identifies where the metric can strengthen evidence of control effectiveness and management oversight.

Table 6
Illustrative regulatory and standards alignment. Applicability depends on the institution, activity, jurisdiction, and specific regulatory perimeter.

Instrument	Relevant theme	AAG contribution
ISO/IEC 27001:2022 [20]	Information-security risk management, controls, continual improvement	AAG provides a quantitative indicator for whether security-review controls are keeping pace with change demand
ISO/IEC 42001:2023 [21]	AI management system, lifecycle governance, monitoring	AAG supplies a software-assurance KPI for AI-enabled engineering processes
ISO/IEC 23894:2023 [22]	AI risk identification and treatment	AAG operationalizes a measurable exposure caused by assurance-capacity imbalance
NIST AI RMF 1.0 [23]	Govern, Map, Measure, Manage	AAG supports measurement and management of AI-related software assurance risk
NIST GenAI Profile [24]	GenAI-specific risk management	AAG complements identification of risks introduced by GenAI use in software lifecycle
NIST SSDF [25]	Secure software lifecycle practices	AAG measures whether required security practices can absorb current demand
NIST SP 800-218A [13]	GenAI/foundation-model secure development	AAG extends the concern to assurance throughput for AI-enabled software systems
DORA [18]	ICT risk and digital operational resilience	AAG provides a forward-looking signal of software-change assurance stress
EU AI Act [19]	Risk-based AI governance, transparency, human oversight	AAG can evidence governance discipline for AI-supported engineering processes where relevant
ADGM/FSRA Cyber Risk Management [27]	Cyber risk embedded in overall risk management	AAG provides a risk-based indicator of whether technology change can be adequately assured
ADGM IT Risk Management Guidance [26,28]	Technology risk identification, controls, resilience	AAG can be integrated into technology-risk metrics and supervisory evidence

4.11.1 ADGM/FSRA Relevance

The ADGM context demonstrates how AAG can be connected to a regulator's risk-based philosophy without pretending that the regulator mandates a specific AAG score. ADGM's IT Risk Management Guidance describes a comprehensive framework for managing technology risk in the financial sector [26]. The FSRA's July 2025 Cyber Risk Management Framework integrated cyber risk management into existing risk frameworks and required compliance from 31 January 2026 [27]. ADGM also states that

firms should tailor cyber controls to vulnerabilities, size, complexity, and reliance on technology [28]. These principles align closely with AAG's emphasis on weighted demand and effective capacity.

The correct interpretation is therefore: AAG is a proposed internal risk metric that could support evidence-based governance in an ADGM-regulated firm; it is not an FSRA-required regulatory ratio unless a future regulatory instrument explicitly adopts it. This distinction should be preserved in any published implementation guide.

4.11.2 Operational Resilience Implications

Operational resilience is concerned with the ability of a financial institution to continue important business services through disruption and recover effectively. Software assurance is one component of that resilience because poorly assured changes can introduce outages, integrity failures, unauthorized access, or delayed recovery. AAG can therefore be used as a leading indicator: rising assurance demand, backlog, and latency may signal that the software-control environment is becoming stressed before a material cyber incident occurs. The metric should not replace resilience testing, incident metrics, recovery objectives, or scenario analysis; it complements them.

5. Discussion

5.1 Why the AAG Concept Matters

The central contribution of AAG is a change in management perspective. Traditional assurance reporting often focuses on pass/fail outcomes: percentage of changes with code review, percentage scanned, number of vulnerabilities open, or number of critical findings. Such metrics can remain green while the underlying workload grows rapidly. AAG adds a capacity dimension. This converts the question from “Did a control run?” to “Did the institution have enough effective control capacity to provide the intended assurance depth?”

This is particularly important with AI because generation and transformation are relatively cheap and fast compared with expert assurance. The institution may be able to generate ten times as many candidate changes without hiring ten times as many security specialists. Automated controls help, but their scope and effectiveness are uneven. The resulting risk is not a single AI vulnerability; it is the possibility that the control system silently becomes under-provisioned.

5.2 AAG Is Not an “AI Is Bad” Metric

The framework is deliberately neutral toward AI adoption. A low-AAG organization may use large amounts of GenAI if its assurance controls scale effectively. Conversely, an organization with little AI adoption may still have a large assurance gap because of technical debt, understaffing, poor architecture, or a surge in conventional development. The “AI” in AAG identifies the driver of the specific change-flow problem under study; it does not make a normative claim about technology.

This distinction also avoids a common policy failure: banning AI because the organization cannot assure it. A better response is to increase observability, restrict the highest-risk use cases, calibrate automated checks, introduce independent review, and manage throughput against assurance capacity. In financial institutions, this risk-based approach is more compatible with operational resilience than blanket prohibition.

5.3 Human Review Should Become Risk-Weighted, Not Merely Mandatory

A requirement that every AI-generated line be manually reviewed can become infeasible at scale and may produce review theater. AAG supports a different approach: all relevant changes should have accountable human ownership, but the depth of review should be proportional to criticality, exposure, and complexity. This is consistent with the practical direction of current secure-AI guidance, which emphasizes human accountability and elevated assurance for security-critical changes [14,15].

The research implication is important. Future studies should measure not only whether a human approved a change, but how much assurance effort was actually spent, which control families were used, whether the reviewer was independent, how often exceptions were granted, and whether risk-weighted sampling improved the detection of material defects.

5.4 Automated Assurance and the Limits of Scanning

Automation is essential for scaling assurance, but the paper treats it as capacity rather than proof of safety. A tool that scans 10,000 changes can still provide weak assurance if it covers only syntax-level patterns, generates excessive false positives, or cannot reason about business logic. AAG therefore recommends “effective” capacity measured against validated assurance outcomes. This principle is especially relevant to financial software, where defects can arise from incorrect authorization, transaction sequencing, settlement logic, key handling, or policy semantics that are not reducible to a single scanner result.

5.5 Relationship to the 2026 Verification Rate Gap

The June 2026 Verification Rate Gap proposal [7] creates an important boundary condition for the novelty claim of AAG. Both constructs recognize that risk can increase when generation outpaces verification. AAG should therefore not be marketed as the invention of that general principle. Its contribution is the operationalization of the principle around security assurance in regulated financial software: a weighted demand model, a measured capacity model, backlog and latency metrics, governance bands, and mapping to finance-sector controls. A future peer-review process may ultimately show that some elements overlap substantially with other frameworks. The paper should welcome that comparison rather than conceal it.

5.6 Practical Benefits

- Security leadership can translate AI-driven engineering growth into a capacity requirement.

- Boards and risk committees can receive a forward-looking assurance indicator rather than only lagging vulnerability counts.
- Engineering teams can identify when security review is becoming a throughput bottleneck before releases are delayed or controls are bypassed.
- Regulated institutions can produce evidence showing how software assurance capacity is aligned with technology-risk appetite.
- Organizations can compare the effect of additional automation, staffing, or process changes using a common metric.

5.7 Limitations

This paper has five principal limitations. First, AAG is conceptual and has not yet been empirically validated. Second, the assurance-unit model requires organizational calibration, and different organizations may define complexity or criticality differently. Third, effective capacity is difficult to estimate because assurance tasks are heterogeneous and some automated controls overlap. Fourth, the provisional risk bands are not normative or regulator-approved. Fifth, the relationship between AAG and actual security outcomes may be nonlinear: a small number of high-criticality changes could matter more than a large volume of low-risk changes. These limitations are not defects to hide; they define the agenda for the next empirical study.

5.8 Future Empirical Validation Study

5.8.1 Study Design

A multi-site longitudinal study should collect three classes of observations over at least six months: software change data, assurance-control data, and security/outcome data. Candidate sources include GitHub/GitLab pull requests, CI/CD logs, SAST/SCA/IaC results, security-review tickets, threat-model records, penetration-test findings, vulnerability-management systems, and change-failure/incident records. AI provenance can be obtained through explicit developer tagging or tool telemetry where permitted by organizational policy.

The key unit of analysis should be the change artifact or normalized change batch. Researchers should compute AAG Ratio, AAG Deficit, backlog, and assurance latency for each period. The study should then examine whether changes in AAG precede changes in security outcomes. Control variables should include team size, system criticality, release frequency, programming language, baseline security maturity, and major architecture changes.

5.8.2 Suggested Variables

Table 7
Proposed variables for empirical validation.

Variable	Type	Example operationalization
AI-assisted throughput	Independent	AI-assisted changes per week / month
AAG Ratio	Independent	AD / AC
AAG Deficit	Independent	max(0, 1 − AC/AD)
Assurance Backlog	Independent	Weighted unassured units carried forward
Assurance Latency	Mediating	Median/p95 hours to assurance completion
Escaped security defects	Dependent	Security issues found after release
Material control exceptions	Dependent	Approved deviations from assurance policy
Change failure / incident rate	Dependent	Operational failures associated with software changes
Security maturity	Control	Baseline security capability score
Automation coverage	Moderator	Effective coverage of automated security controls

5.8.3 Validation Strategy

The first validation test should assess construct validity: do AAG units correlate with actual review effort and expert-rated complexity? The second should assess predictive validity: does sustained AAG correlate with rising assurance latency or escaped defects? The third should assess calibration: which risk weights and thresholds best discriminate between periods with and without material security outcomes? The fourth should assess intervention efficacy: when the organization increases human review, automation, or release gating, does AAG decline and do outcome measures improve?

A mixed-effects regression can account for differences among organizations or teams, while survival or time-to-event models can test whether high AAG periods are associated with faster occurrence of material security events. A queueing or time-series model could estimate whether the system stabilizes when capacity growth matches demand growth. The empirical study should preregister key hypotheses and exclude post-hoc threshold selection from the primary analysis.

5.8.4 Data Governance and Ethics

Because software repositories and security tickets can contain sensitive operational information, a production study should use data minimization, de-identification, controlled access, and organization-approved telemetry. No customer data is required to validate the core construct. Researchers should be especially careful not to publish exact vulnerabilities, secret material, or exploitable infrastructure configurations. AAG is fundamentally a management metric, so the study can often be performed using aggregated or pseudonymized records.

6. Conclusions

Generative AI is changing software delivery faster than many traditional assurance processes were designed to change. Evidence from controlled experiments and field studies shows meaningful increases in developer throughput [1–3]. At the same time, security research shows that AI-assisted or AI-generated code can contain vulnerabilities and can affect developer confidence and review behavior [4–6,11,12]. Regulatory and standards bodies are simultaneously increasing emphasis on AI governance, technology risk, secure development, and resilience [13,16,18–28]. The strategic problem is therefore not simply AI quality; it is whether the security assurance system remains proportionate to the volume, complexity, and criticality of the changes it must assure.

This paper introduced the AI Assurance Gap (AAG) as a conceptual framework for making that problem measurable. The AAG Ratio compares risk-weighted assurance demand with effective assurance capacity. AAG Deficit expresses the uncovered proportion of demand, while Assurance Backlog and Assurance Latency capture the accumulating stock and delay that flow metrics can hide. Together, the measures create a bridge between engineering throughput and security governance.

The paper deliberately avoids claiming empirical validation or regulator endorsement. The proposed weighting coefficients and threshold bands are initial research constructs that require testing. The novelty claim is also deliberately scoped in light of the June 2026 Verification Rate Gap proposal: AAG is a specialized security-assurance operationalization for regulated financial software delivery, not a claim to have discovered the general principle that generation can outrun verification [7].

The most important future work is therefore straightforward: collect real organizational data, calibrate assurance units, measure effective control throughput, and test whether AAG predicts assurance stress or security outcomes. If validated, AAG could give security and risk leaders a practical way to answer a question that conventional control checklists struggle to answer: not whether security controls exist, but whether the organization has enough assurance capacity for the software it is now able to create.

Abbreviations

AAG
AI Assurance Gap
AAGR
AAG Ratio
AD
Assurance Demand
AC
Assurance Capacity
AB
Assurance Backlog

AL
Assurance Latency
AI
Artificial Intelligence
GenAI
Generative Artificial Intelligence
IaC
Infrastructure as Code
SAST
Static Application Security Testing
DAST
Dynamic Application Security Testing
SCA
Software Composition Analysis
SDLC
Software Development Life Cycle
DORA
Digital Operational Resilience Act
FSRA
Financial Services Regulatory Authority
ADGM
Abu Dhabi Global Market
NIST
National Institute of Standards and Technology
OWASP
Open Worldwide Application Security Project.

Declarations

Note: This manuscript presents a conceptual framework. Provisional coefficients and thresholds are research constructs and are not regulatory requirements or empirically validated risk limits.

Author Contributions: Conceptualization, methodology, investigation, formal analysis, writing—original draft preparation, writing—review and editing, visualization, and project administration were performed by S.H.R. The author has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable because this conceptual study did not involve human participants, animals, or identifiable personal data.

Informed Consent Statement: Not applicable.

Data Availability Statement: No external datasets were used or analysed in this conceptual study. The hypothetical numerical values used in the illustrative case study were generated solely to demonstrate the proposed AI Assurance Gap (AAG) framework and are included in the manuscript. No real-world or proprietary datasets were collected or analysed.

Acknowledgments: During the preparation of this manuscript, the author used OpenAI ChatGPT (GPT-5.6 Luna) for conceptual development, literature-organization assistance, structural editing, and language drafting. The author reviewed and edited the output and takes full responsibility for the content of the publication.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Peng, S.; Kalliamvakou, E.; Cihon, P.; Demirer, M. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv 2023, arXiv:2302.06590. https://doi.org/10.48550/arXiv.2302.06590.
2. Cui, K.Z.; Demirer, M.; Jaffe, S.; Musolff, L.; Peng, S.; Salz, T. The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers. Management Science 2026. https://doi.org/10.1287/mnsc.2025.00535.
3. Gambacorta, L.; Qiu, H.; Shan, S.; Rees, D.M. Generative AI and labour productivity: A quasi experiment on coding. Journal of Financial Stability 2026, 84, 101543. https://doi.org/10.1016/j.jfs.2026.101543.
4. Pearce, H.; Ahmad, B.; Tan, B.; Dolan-Gavitt, B.; Karri, R. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In Proceedings of the 43rd IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 22–26 May 2022; pp. 754–768. https://doi.org/10.1109/SP46214.2022.9833571.
5. Fu, Y.; Liang, P.; Tahir, A.; Li, Z.; Shahin, M.; Yu, J.; Chen, J. Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study. ACM Transactions on Software Engineering and Methodology 2025, 34. https://doi.org/10.1145/3716848.
6. Kudriavtseva, A.; Hotak, N.A.; Gadyatskaya, O. My Code Is Less Secure with Gen AI: Surveying Developers' Perceptions of the Impact of Code Generation Tools on Security. In Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing, Catania, Italy, 31 March–4 April 2025; pp. 1637–1646. https://doi.org/10.1145/3672608.3707778.
7. Gorai, J. The Verification Rate Gap: A Conceptual Metric for AI Development Risk. SSRN 2026. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=6904238 (accessed on 8 October 2026).
8. Fraser, K.; Sergeyuk, A.; Zakharov, I. Our Framework for Reviewing AI-Generated Code. JetBrains Research Blog, 2026. Available online: https://blog.jetbrains.com/research/2026/10/review-ai-generated/ (accessed on 8 October 2026).

9. Trebacz, M.; Arnesen, S.; Cassirer, A.; Johnson, M.; Lin, X.; Sottiaux, T. A Practical Approach to Verifying Code at Scale. OpenAI Alignment Research, 2025. Available online: https://alignment.openai.com/scaling-code-verification/ (accessed on 8 October 2026).
10. Barke, S.; James, M.B.; Polikarpova, N. Grounded Copilot: How Programmers Interact with Code-Generating Models. Proceedings of the ACM on Programming Languages 2023, 7, OOPSLA1, 85–111. https://doi.org/10.1145/3586030.
11. Perry, N.; Srivastava, M.; Kumar, D.; Boneh, D. Do Users Write More Insecure Code with AI Assistants? In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, Copenhagen, Denmark, 26–30 November 2023. https://doi.org/10.1145/3576915.3623157.
12. Pandey, M.; Kumar, S. Secure Code Generation With Open Source Generative AI Models. IEEE Software 2026, 43, 76–86. https://doi.org/10.1109/MS.2026.3672189.
13. Booth, H.; Ogata, M.; Vassilev, A.; Stanley, M.; Scarfone, K.; Souppaya, M. Secure Software Development Practices for Generative AI and Dual-Use Foundation Models: An SSDF Community Profile. NIST SP 800-218A, 2024. https://doi.org/10.6028/NIST.SP.800-218A.
14. OWASP Foundation. Secure Coding with AI Cheat Sheet. OWASP Cheat Sheet Series, 2026. Available online: https://cheatsheetseries.owasp.org/cheatsheets/Secure_Coding_with_AI_Cheat_Sheet.html (accessed on 8 October 2026).
15. OWASP Foundation. Artificial Intelligence Security Verification Standard (AISVS), Version 1.0, 2026. Available online: https://owasp.org/www-project-artificial-intelligence-security-verification-standard/ (accessed on 8 October 2026).
16. Financial Stability Board. Sound Practices for Responsible Adoption of Artificial Intelligence (AI): Consultation Report. 10 June 2026.
17. Bank of England. Financial Stability in Focus: Artificial Intelligence in the Financial System. 9 April 2025.
18. European Parliament; Council of the European Union. Regulation (EU) 2022/2554 of 14 December 2022 on Digital Operational Resilience for the Financial Sector (DORA). Official Journal of the European Union, 27 December 2022.
19. European Parliament; Council of the European Union. Regulation (EU) 2024/1689 of 13 June 2024 Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act). Official Journal of the European Union, 12 July 2024.
20. ISO/IEC. ISO/IEC 27001:2022 Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements. ISO: Geneva, Switzerland, 2022.
21. ISO/IEC. ISO/IEC 42001:2023 Information Technology—Artificial Intelligence—Management System. ISO: Geneva, Switzerland, 2023.
22. ISO/IEC. ISO/IEC 23894:2023 Information Technology—Artificial Intelligence—Guidance on Risk Management. ISO: Geneva, Switzerland, 2023.
23. National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1, 2023. https://doi.org/10.6028/NIST.AI.100-1.

24. National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. NIST AI 600-1, 2024.
25. Scarfone, K.; Souppaya, M.; Dodson, D. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. NIST SP 800-218, 2022. https://doi.org/10.6028/NIST.SP.800-218.
26. Financial Services Regulatory Authority of Abu Dhabi Global Market. Information Technology Risk Management Guidance. ADGM, 20 November 2024.
27. Financial Services Regulatory Authority of Abu Dhabi Global Market. ADGM's FSRA Issues Cyber Risk Management Framework. 29 July 2025. Compliance required from 31 January 2026. Available online: https://www.adgm.com/media/announcements/adgms-fsra-issues-cyber-risk-management-framework (accessed on 8 October 2026).
28. Financial Services Regulatory Authority of Abu Dhabi Global Market. Governance Principles and Practices to Mitigate Cyber Threats and Crime. ADGM, 2020. Available online: https://www.adgm.com/documents/legal-framework/guidance-and-policy/fsra/adgm-governance-principles-and-practices-to-mitigate-cyber-threats-and-crime.pdf (accessed on 8 October 2026).
29. National Institute of Standards and Technology. Cybersecurity Framework (CSF) 2.0. NIST CSWP 29, 2024.
30. Liu, C.L.; et al. The Impact of AI Tool on Engineering at ANZ Bank: An Empirical Study on GitHub Copilot within Corporate Environment. arXiv 2024, arXiv:2402.05636.
31. Benetti, A.; Filannino, M. GitHub Copilot: A Systematic Study. In Proceedings of Ital-IA 2024, CEUR Workshop Proceedings, 2024; Volume 3762, pp. 6–11.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- AppendixAB.docx
- TableS1.docx