Blue Trail Software

The Engineering Leadership Brief

The Code Passed. Can You Prove It's Ready To Deliver?

The AI Verification Gap: what research reveals about code quality, delivery risk, and the evidence engineering leaders need before scaling AI-assisted development.

sales@bluetrail.software (415) 247 - 9503 www.bluetrail.software

The Code Passed.
Can You Prove It's Ready To Deliver?

AI adoption is only the starting point. Proving its engineering impact is harder.

Engineering leaders are under pressure to show that AI-assisted development improves the path from a code change to a dependable release. They must demonstrate meaningful delivery gains while protecting software quality, security, maintainability, and release confidence. A coding assistant may shorten implementation; the organization still has to establish that each change meets its requirements, behaves correctly at system boundaries, and can be supported in production.

If code production increases but review queues, regression effort, rework, and release delays remain stubborn, the work may simply be accumulating later in the delivery path. That pattern is not proof that AI caused the problem. It is a reason to investigate where effort is going before buying more tools, expanding automation, or adding QA capacity.



These survey results describe respondents' experiences; they do not predict what will happen in a particular company. Taken together, they point to a practical leadership task: distinguish useful acceleration from the additional work needed to verify, debug, and safely deliver the output.

What the research shows - and what it cannot tell you

The studies do not produce one universal verdict on AI-assisted software development. They examine different tasks, teams, time periods, and outcomes: individual task speed, security properties, review behavior, or delivery performance. Their value is to help leaders ask better questions of their own systems, not to substitute someone else's result for internal evidence.

- **AI use is widespread; verification remains part of the work**
  In Stack Overflow's 2026 Developer Survey, 83% of respondents said they currently use at least one AI tool in their work. When asked what they do before using AI-generated output, 76.5% said they run it locally, 63.9% compare it against the existing codebase, and 53% inspect tests or security implications. Only 9.6% said they use AI-generated output as-is. [1]
  These are self-reported survey responses, not measurements of the time required to verify code or proof that the checks are effective. They do, however, show that developers commonly take additional steps before using AI output. Engineering leaders need internal evidence about where that work occurs, how much effort it consumes, and whether their verification process provides sufficient confidence without eroding the expected productivity gains.

- **A faster coding step may not shorten the delivery path**
  DORA's 2024 research associated AI use with higher individual productivity while also reporting negative relationships with delivery throughput and stability. Its 2025 report argues that results depend heavily on the engineering practices and organizational conditions surrounding AI, rather than on tool adoption alone. [2][3]
  For leaders, the useful comparison is not only how quickly a developer completes an implementation task. It is whether the organization can review, verify, integrate, and release the resulting change with less total waiting, rework, or instability.

- **Productivity findings depend on the task and setting**
  A 2025 randomized controlled trial from METR found that experienced open-source developers working in repositories they knew well took 19% longer on the selected tasks when using the AI tools evaluated in that study. METR explicitly cautioned against generalizing the result to most developers or all types of software work. [4]
  METR's February 2026 update described its later experiment as difficult to interpret because of selection effects and unreliable time measurements for some developers using multiple agents. The researchers said the findings offered weak evidence about the size of the productivity effect. The lesson is methodological: tool use and work patterns are changing, so productivity claims need carefully matched tasks and credible measurement. [5]

What this means for your organization

- **Do not assume AI is automatically speeding up delivery - and do not assume one study proves that it is slowing your team down. Establish a baseline, compare like-for-like work, and measure the full path from change creation to verified release**

Passing tests, secure code, and misleading comparisons

- **Functional success is only one part of assurance**
  Veracode's 2026 GenAI Code Security Report evaluated AI-generated code across multiple models, languages, and deliberately designed security tasks. Approximately 44% of the tested code-generation tasks produced code containing a detectable OWASP Top 10 vulnerability, while the average security pass rate was approximately 56%. [6]
  This is a controlled, vendor-led benchmark using raw model outputs, not an estimate that 44% of AI-generated production code is insecure. The findings highlight an important distinction: syntactic correctness does not establish security, and code that passes functional checks may still contain vulnerabilities. Engineering teams need verification controls that assess the risks relevant to each change, not just whether the code runs.

- **Large-scale comparisons can hide differences in risk**
  An ESEM 2026 study examined AI-suggested code across more than 400,000 diffs from over 10,000 developers at Meta. The researchers observed lower review time and lower production-severity-event rates for diffs containing AI code, but both relationships were confounded by that code appearing more often in less central parts of the codebase. [8]
  That context changes how the result should be read. A small change in a test file and a change to a core authentication service have different consequences and review needs. A comparison that treats them as equivalent can make a process look better or worse for reasons unrelated to the quality of the AI assistance itself.

- **Trust depends on the task and the evidence available**
  In Stack Overflow's 2026 survey, 48% of respondents said they trusted AI output when they could easily verify it; 6.6% said they trusted it for many tasks, including important work decisions. [7] This is not an argument against using AI. It suggests that confidence is conditional: the team needs a way to check the output in proportion to the consequences of getting it wrong.

The conclusion is not “AI code is bad.”

- **The evidence is more useful than that:** adoption, output, test pass rates, security, review time, and delivery outcomes measure different things. They should not be treated as interchangeable proof of engineering performance.

The Code Passed.
Can You Prove It's Ready To Deliver?

The Blue Trail Software Verification Evidence Chain

Production signal: What is AI changing? Map AI-assisted changes through the workflow and examine their volume, size, complexity, affected areas, and risk - not AI usage alone.
Verification signal: What evidence supports acceptance? Examine PR queue and rework, approval and validation cycles, CI/CD quality gates, risk coverage, test reliability, and unresolved findings.
Delivery signal: What happens after acceptance? Track end-to-end delivery time, release predictability, escaped defects, rollbacks or incidents, and repeated rework.
The gap to investigate is the disconnect between these signals: more AI-assisted output without clear evidence that review and verification can keep pace or that delivery outcomes are improving. Blue Trail Software uses this chain to structure the audit; it is a practical diagnostic lens, not an industry standard or validated benchmark.

Six questions every engineering leader should be able to answer

Use these questions to structure an internal review. They are not a maturity score, and a single yes/no answer rarely explains a complex delivery system. The goal is to identify what the team can demonstrate, what is still unknown, and where evidence should be collected next.

01
Where does elapsed time accumulate?
Track time from pull request creation to first review, approval, test completion, rework, and release. Separate waiting time from active work. A faster coding step may not shorten the full cycle.

02
Are changes growing harder to verify?
Look beyond the number of changes. Examine change size, files touched, dependencies affected, code criticality, and how often a change needs another review or test cycle.

03
Do tests give evidence about the risks that matter?
Review whether high-risk paths, integration boundaries, failure handling, permissions, and security-sensitive behavior are exercised. Coverage is a useful signal, not a complete measure of test effectiveness.

04
How much QA capacity is lost to noisy signals?
Measure flaky tests, repeated failures, false alarms, test-environment instability, and manual triage. Unreliable feedback consumes time and can make genuine failures easier to miss.

05
Are defects and rework changing as AI use expands?
Trend escaped defects, rollback or incident rates, re-opened issues, and rework by product area and change type. Compare periods and teams carefully: do not attribute every change to AI without controlling for other factors.

06
Can you identify the bottleneck with evidence?
Can the team show where verification is delayed, what kinds of changes drive that delay, and whether the constraint is test design, execution, review, environments, defect diagnosis, or release policy? If not, visibility may be the first problem to solve.

A practical rule

- **If the only evidence of improvement is more AI use, more generated code, or more tests, the team has not yet demonstrated the impact on delivery. Connect engineering activity to verification effectiveness and production outcomes.**

Turn published research into an internal diagnosis

External research can help frame the investigation, but it cannot identify the constraint inside a specific engineering organization. The answer depends on the architecture, change risk, workflow, verification strategy, and telemetry available in that environment.

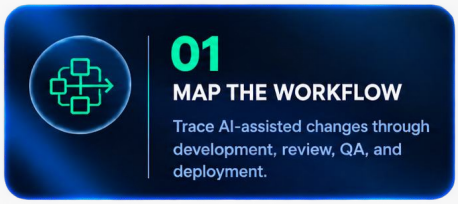
The Code Passed.
Can You Prove It's Ready To Deliver?

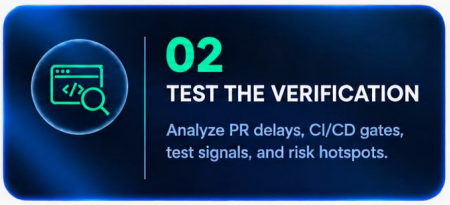
Signals that warrant a closer look

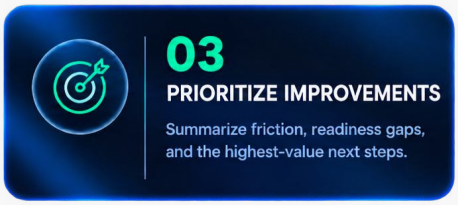
AI-assisted development is expanding, but end-to-end delivery time or predictability is not improving.
Pull requests are becoming larger or waiting longer for review and approval.
The test suite is growing, but the team cannot show which important failure modes it detects reliably.
Regression, test-environment setup, flaky results, or defect triage repeatedly pushes releases back.
The organization cannot relate changes in AI use to verification effort, rework, production stability, or delivery results.

How Blue Trail Software can help

- Blue Trail Software (BTS) is a nearshore software engineering partner for complex digital products, with capabilities across product engineering, AI/GenAI, and QA/testing. For teams adopting AI coding tools, BTS connects workflow visibility, review and CI/CD evidence, and delivery outcomes to help locate where verification is adding friction or leaving important risks unexamined.
  The AI Verification Bottleneck Audit is a focused, two-week diagnostic for engineering teams experiencing hidden delivery bottlenecks, review slowdowns, or quality risks as AI-assisted development expands. It maps how AI-generated code moves through development, review, QA, and deployment, then investigates the delays and risk signals that shape release confidence. The goal is to identify the most consequential constraint and prioritize practical improvements - not to deliver a generic maturity presentation.







Inside Blue Trail Software's two-week AI Verification Bottleneck Audit

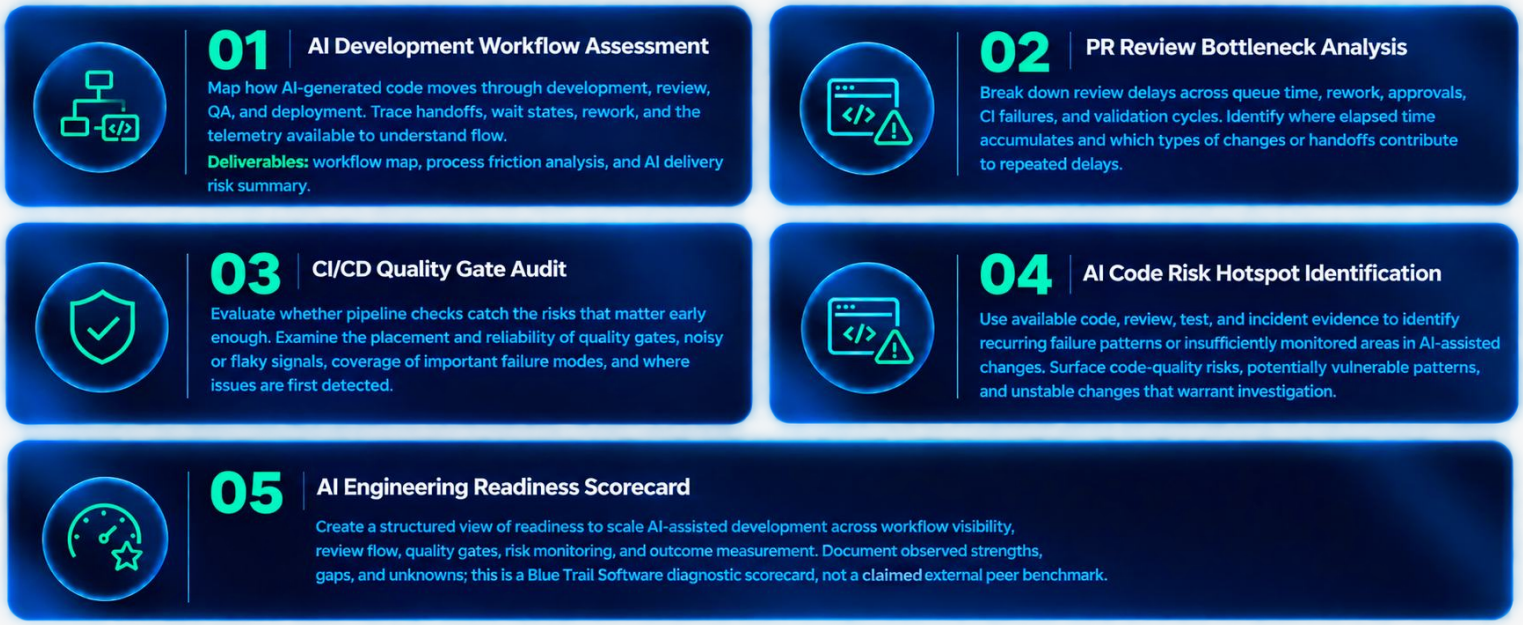
- *For engineering teams adopting AI coding tools and struggling with hidden delivery bottlenecks, review slowdowns, or quality risks.*

AI can shorten the time required to produce a code change, but review, validation, and CI/CD still have to establish whether that change is ready. Without visibility into AI-assisted workflow patterns, teams may miss recurring defects, vulnerabilities, unstable changes, or costly feedback delays. Blue Trail Software examines the engineering system around the tools - not simply whether AI is being used.

The five connected workstreams

The audit moves from workflow visibility to verification evidence and prioritized action. Its findings depend on the agreed scope, available telemetry, and access to relevant workflow artifacts.

The Code Passed.
Can You Prove It's Ready To Deliver?



Audit Workstream | What Blue Trail Software Examines

What the team takes away
A prioritized view of where delivery friction occurs, which verification risks deserve attention, and what to improve first. Core outputs include the workflow map, process friction analysis, AI delivery risk summary, PR/CI findings, AI Engineering Readiness Scorecard, and practical next steps. Findings are tied to available evidence and the agreed audit scope.

The business outcomes Blue Trail Software is working toward
Reduce delivery bottlenecks by identifying the review, validation, or pipeline constraints slowing engineering flow.
Improve AI code reliability by surfacing recurring failure patterns and verification gaps that merit attention.
Increase team visibility into review cycles, rework, quality-gate performance, and AI workflow efficiency.
Build safer AI development processes while protecting the delivery speed teams are trying to gain.

Turn the report into a diagnosis of your workflow
Published research can show why these questions matter. The two-week AI Verification Bottleneck Audit examines where they show up in your engineering environment. Bring Blue Trail Software one recurring release delay, review queue, CI failure pattern, or difficult change as a starting point.

Request an AI Verification Bottleneck Audit
Talk with Blue Trail Software's Head of QA about your verification constraints, the evidence available in your workflow, and whether the focused audit is the right next step.

AI VERIFICATION BOTTLENECK AUDIT

Find the constraint. Understand the evidence. Decide what to improve first.

Discuss an audit: bluetrail.software/contact-us

DORA / Google Cloud. (2024). "Accelerate State of DevOps Report 2024." https://dora.dev/research/2024/dora-report/

References and interpretation notes

This brief combines developer surveys, organizational research, a randomized productivity study, a controlled security benchmark, and a large-scale empirical study. These sources measure different outcomes; the percentages should not be combined or treated as a single comparable dataset.

- *Stack Exchange Inc. (2026). "AI tool usage" and "Next steps after receiving an AI-generated answer for a development task." Stack Overflow Developer Survey 2026. https://survey.stackoverflow.co/2026/ai/data/ai-select and https://survey.stackoverflow.co/2026/ai/data/ai-work-next. AI-tool adoption and self-reported verification practices; survey data licensed under ODbL 1.0.*
- *DORA / Google Cloud. (2024). "Accelerate State of DevOps Report 2024." https://dora.dev/research/2024/dora-report/ Reports benefits and trade-offs associated with AI use and delivery performance.*
- *DORA / Google Cloud. (2025). "The State of AI-assisted Software Development." https://dora.dev/research/2025/dora-report/ Discusses how organizational practices and conditions shape the results teams get from AI-assisted development.*
- *Becker, J., Rush, N., Barnes, E., & Rein, D. (2025). "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity." METR. https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/ Randomized controlled trial in a specific developer/task setting; authors caution against overgeneralization.*
- *Becker, J., Rush, N., Cunningham, T., Rein, D., & Mahamud, K. (2026). "We are Changing our Developer Productivity Experiment Design." METR. https://metr.org/blog/2026-02-24-uplift-update/ Explains selection effects and measurement limitations in later experiments.*
- *Stack Exchange Inc. (2026). "Trust in AI tools," Stack Overflow Developer Survey 2026. https://survey.stackoverflow.co/2026/ai/data/ai-trust Trust responses from 14,304 respondents; licensed under ODbL 1.0.*
- *Mockus, A., Rigby, P. C., Stewart, D., Maddila, C., & Nagappan, N. (2026). "A Preliminary Analysis of the Impact of AI Assisted/Generated Code on Quality, Centrality and Review Time at Scale." ESEM 2026, LIPIcs 394, Article 93. https://doi.org/10.4230/LIPIcs.ESEM.2026.93 Analysis of more than 10,000 developers and 400,000 code diffs at Meta; associations are context-dependent.*

How to read these findings

Survey responses reflect what participants report. Observational relationships do not necessarily establish causation. Controlled benchmarks depend on their selected tasks and test conditions. Results from a single organization or task population may not generalize to other architectures, teams, or tools. This brief uses the studies to identify questions worth asking - not to claim that every organization has the same problem. The Blue Trail Software Verification Evidence Chain and the practical questions in this brief are Blue Trail Software's synthesis for guiding an investigation; they are not published research findings or a validated industry index.

ABOUT BLUE TRAIL SOFTWARE | bluetrail.software

- Blue Trail Software is a California Benefit Corporation founded in 2013 and a nearshore software engineering partner for complex digital products. Its capabilities include product engineering, AI/GenAI, QA/testing, embedded/IoT, and related engineering disciplines.