aspire SYSTEMS

Whitepaper

The AI Acceleration Gap

Author

Jothi Rengarajan

Head of Gen AI Practice, Aspire Systems

www.aspiresys.com

Why AI tools deliver lower than expected, and what it takes to reach that expected level?

10–15%	50–75%+	4×
Real-world acceleration without a framework	Acceleration with a structured approach	Difference between the two

Source: Aspire Systems engagement data, 2023–2026.

Executive Summary

The software engineering sector is witnessing a major disconnect: AI coding tools are accelerating software development, but software delivery is comparatively slower. The key concern is that while developers report higher throughput, their release velocity remains constrained by slower reviews, rising defects, increasing code churn, and growing operational costs.

This whitepaper argues that the gap is structural, not technological. Consolidating research and first-hand experience, this paper answers pressing questions, like why the delivery constraints persist, and introduces a structured framework – PAVE (Progressive AI ValuE) to close the gap.

Introduction: The Shift in SDLC Bottlenecks

Engineering leaders managing large-scale delivery ecosystems share a common pursuit: velocity of the software development lifecycle. Yet, modern software engineering organizations are experiencing a widening disconnect. Individual development tasks are accelerating meaningfully, while overall product release cycles remain largely unchanged.

Faros AI's 2026 Engineering Report, drawing on two years of telemetry from 22,000 developers across more than 4,000 teams, tracked each organization's own metrics between its periods of lowest and highest AI adoption and found real, substantial throughput gains: epics completed per developer up 66%, task throughput per developer up 33.7%, and pull request merge rate per developer up 16.2%. These numbers represent genuine, measurable delivery acceleration at the individual and team level.

But the same dataset shows why that acceleration is not reaching the program level intact. Across the same population, bugs per developer rose 54%, production incidents per pull request roughly tripled, code churn – the rate at which recently written code is rewritten or reverted – grew tenfold, and median pull request review time increased roughly fivefold. The report's own framing is direct: across every downstream stage of the SDLC, the signal is consistent — volume is up, quality is down, and the gap between the two is widening as adoption deepens.

The study is clear evidence that AI is not failing to accelerate work. It is accelerating work faster than review, testing, and release systems can safely absorb.

The Promise vs. Expected Gap

Every engineering leader has encountered the same narrative: AI-assisted development delivers transformational productivity gains. And the underlying task-level numbers, as discussed in the last section, confirms AI's inherent powers. ***METR's most recent survey*** *even reports an increase in the developers' reliance on AI assistants. But the promise of 'transformational gains' still remains questionable.*

There's no denying that organizations are attracted to the 'promise'. By 2026, adoption of AI coding assistants has reached 97% across enterprise software development organizations, per **Black Duck's State of AI-Powered Software Development survey** of 831 enterprise engineers and DevOps professionals conducted with independent research firm UserEvidence. Engineers are, at this point, almost universally working with GitHub Copilot, Claude Code, Cursor, or an equivalent.

But then, despite the excitement and initial investments, six months later, the program is running just 12% faster – closer to the 10–15% ceiling – than to the 55% headline promise. Not even the 33.7% task-level figure the Faros AI's telemetry showed.

In such situations, the tools are normally blamed, sceptics feel vindicated, and budgets are questioned. **MIT's Project NANDA** – based on 52 executive interviews, a survey of 153 business leaders, and analysis of 300 public AI deployments – found that 95% of enterprise generative AI pilots delivered no measurable profit-and-loss impact, with only 5% of integrated pilots producing significant value, despite an estimated $30–40 billion in enterprise investment. A gap of this size means that AI alone does not automatically create an AI-enabled engineering organization.

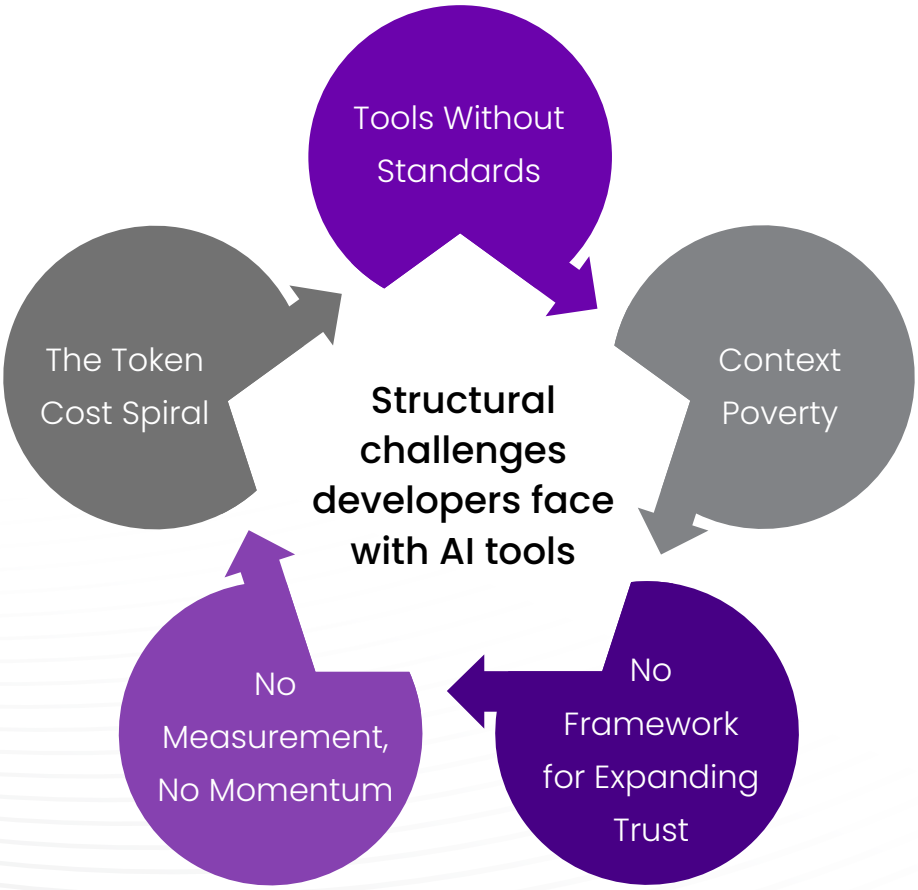
The difference between the outcome most programs see and the promise most vendors make is not the tool – it is the operating model surrounding it: how teams use it, how knowledge is structured, how quality is protected, and how progress is measured.

The Gap is Real, and it is Structural

The 10–15% ceiling is not an anomaly confined to poorly run programs. It is a pattern documented consistently across the independent research.

DevOps Research and Assessment **(DORA)'s 2025 State of AI-assisted Software Development** report shows a cluster analysis that identifies seven distinct team profiles ranging from "harmonious high-achievers" to teams caught in a "legacy bottleneck," demonstrating that outcomes with identical AI tools diverge sharply based on organizational practice rather than tool capability.

That divergence traces back to five structural conditions that AI tools cannot resolve on their own.



1. Tools Without Standards

AI tools are adopted by individuals, not by programs, and the resulting variance in outcomes is measurable. The 2025 DORA report identifies what its authors term a "trust paradox": only 24% of respondents report a great deal or a lot of trust in AI-generated output, while 30% report little to no trust – a split that correlates closely with the uneven, individually determined usage patterns DORA's team-profile analysis surfaces.

Without shared standards governing what context to provide, what output to validate, and what to accept versus rework, program-level acceleration becomes as variable as individual technique.

2. Context Poverty

AI models generate output in proportion to the quality of context available to them, and the cost of insufficient context is now quantifiable at scale. **GitClear's 2025 AI Copilot Code Quality** report, analyzing 211 million lines of changed code across 2020–2024, found that refactored or moved code fell from 24.1% of changed lines in 2020 to 9.5% in 2024, copy-pasted code exceeded reused code for the first time ever, and code cloning increased fourfold. The same analysis found that code churn – the share of newly written code revised within two weeks – rose from 3.1% in 2020 to 5.7% in 2024, indicating a growing volume of premature or poorly contextualized commits.

Without structuring domain knowledge, documenting module boundaries, and establishing coding conventions legible to AI systems, output quality plateaus regardless of which tool is deployed.

3. No Framework for Expanding Trust

While software licenses are visible, the ongoing effort required to verify AI-generated code for accuracy, reliability, and security is often overlooked. **Sonar's State of Code Development Survey** report studied over 1,100 developers and found that while 96% of developers don't have complete trust in the AI code output, only 48% of them verified the code before committing to it.

> *Without a framework for expanding trust on the basis of tracked evidence, organizations default to one or two failure modes: over-review, in which every output is treated as suspect, and the velocity gain evaporates; or under-review, in which quality debt accumulates invisibly.*

4. No Measurement, No Momentum

A major concern is that when teams layer AI over existing process dysfunction, they expect resolution. With 98% more pull-request rates, per **Faros AI's** study, it is logical that existing dysfunctions will only intensify, not resolve. If code review is already a bottleneck, increased change volume and frequency will produce longer delays.

> *Without dashboards tracking AI-specific acceleration and defect rates at the workstream level, there is no feedback loop through which effective patterns are identified, and failures are diagnosed.*

5. The Token Cost Spiral

As AI is billed by tokens, adoption introduces a cost dimension extending from the engineering team to the boardroom, with costs scaling directly with usage. The **FinOps Foundation's 2026 State of FinOps** report found that 73% of enterprises reported AI costs exceeding original projections, and **Ramp** reports that enterprise AI token consumption increased roughly thirteenfold since January 2025 – growth driven by unmanaged consumption patterns rather than a proportional increase in users. A key reason is that organizations frequently default every task, including simple ones, to the most capable and expensive model available because model selection is typically invisible to end users and no routing logic matches task complexity to model cost.

> *Without structured oversight, the majority of enterprise AI spending today remains unmanaged.*

Across all five conditions, the pattern is consistent: the constraint on AI-driven acceleration is the absence of a governing framework around them.

The Cost of Accepting the 10% Ceiling

Leaders often regard the 10–15% ceiling as a disappointing but tolerable outcome. The figures below reflect Aspire Systems' own engagement data across 100+ programs, shown here to illustrate the order of magnitude at stake.

Metric / Asset	At 10–15% Acceleration	At 50–75% Acceleration
Team of 50 engineers	Equivalent of ~6 extra engineers	Equivalent of 25–37 extra engineers
$5M delivery budget	~$500K in recovered value	$2.5M–$3.5M in recovered value
12-month program	Same timeline, modest velocity lift	Compresses 12 months into 7–9 months
Defect rates	Unchanged or marginally improved	Materially fewer design-to-code defects, per Aspire program data
AI token spend	Untracked, inflated by inefficiency	Standardized and optimized per task type

Source: Aspire Systems engagement data, 2023–2026, across 100+ program deployments.

Six Hard-Won Lessons by Aspire Systems

During the program deployments, which included 50+ GitHub Copilot engagements and 30+ Claude Code programs, Aspire Systems realized six primary principles that can separate programs that break through the 10% ceiling from those that do not:

Lesson 1 **Acceleration is a program capability, not an individual one.**

Outcomes vary by team profile and organizational practice far more than by which AI tool is deployed. The program, not the individual, is the unit of AI capability.

Lesson 2 **Context investment pays back every sprint.**

Programs that invest early in making their codebase legible to AI consistently outperform those deploying AI directly onto unprepared codebases.

Lesson 3 **Trust must be earned, not granted.**

AI autonomy expanded on tracked evidence, such as completion rates, defect rates, and review coverage, sustains quality gains; autonomy granted by decree does not.

Lesson 4 **Measurement is the engine of compounding.**

Programs that track AI-specific metrics sprint by sprint create the feedback loop that surfaces what works.

Lesson 5 **Maturity advances in stages, not all at once.**

Attempting full AI autonomy before foundations are in place is the most common cause of AI adoption failure. AI magnifies whatever organizational maturity already exists, for better or worse.

Lesson 6 **Standardization is the most powerful cost-control lever.**

Standardization is not merely a quality mechanism. It is the financial model that makes large-scale AI adoption viable.

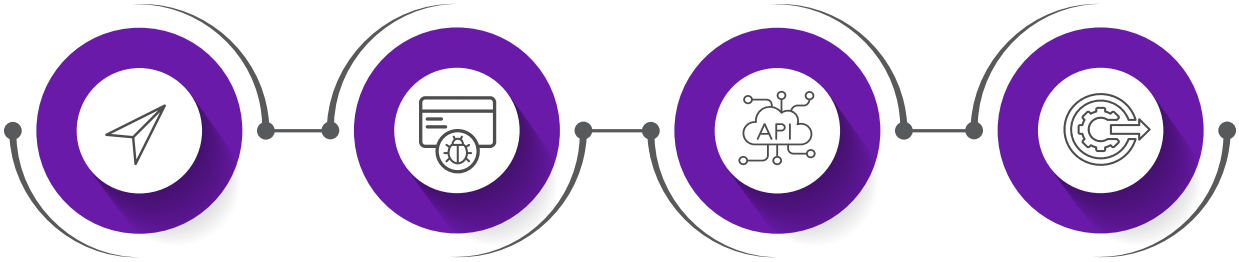
PAVE – A Better Way

These six lessons form the foundation of Aspire Systems' structured methodology: **PAVE (Progressive AI ValuE).**

PAVE defines how AI agents are standardized and governed, how trust is progressively extended based on tracked evidence, how context is engineered, and how AI consumption is standardized so token costs scale with value rather than with individual inefficiency.

It is a framework – a way of operating that compounds AI's gains.

PAVE's approach enables organizations to achieve:



50% faster SDLC delivery

Materially fewer design-to-code defects

Faster API and front-end shipment

Meaningfully lower AI token spend per unit of output

PAVE (Progressive AI ValuE): Progressive AI Acceleration Excellence

Each stage unlocks capabilities, shifts the human-AI balance, and requires a gate to be cleared before advancing

Spark	Shift	Delegate	Orchestrate	Autonomous
10–15% gain	*25–40% gain*	*40–55% gain*	*55–75% gain*	*75%+ gain*

M0 — M1 — M2 — M3 — M4

Gate 1

- Standardized Agents and Skills
- Clear Context Sources and Engineering

Governance + standards

- Agent Metrics and Tracking
- AI-centered SDLC
- AI literacy training for all engineers

Gate 2

- Human escalation framework
- Agent KPI tracking & optimization
- Security & compliance review for agent permissions

Gate 3

- Agent role & Skill refinement framework
- Incident response & liability framework
- Agent Collaboration Governance

Gate 4

- Trust + compliance

AI Assisted - Engineer Done	Engineer Supervised	Task - AI Completed and Engineer Reviewed	Flow AI Completed With Human Gates	AI Flow Completed End to End With Aito Gate

Aspire Proprietary Platform – SoftSpell | Aspire Partnership – Claude Code, GitHub Copilot, Cursor

HUMAN ↔AI BALANCE SHIFT

● Human ● AI Agent

Overview of Pave Framework

In Aspire Systems' PAVE, programs move through distinct maturity levels – from M0 through to M4 – each with a defined human-AI balance and measurable acceleration band, beginning where they are today and advancing as evidence accumulates.

- **Early stages** establish standards, context, governance, measurement, and token cost baselines.
- **Middle stages** delegate tasks and workflows to AI, with structured human review and optimized prompt patterns.
- **Advanced stages** move to AI-orchestrated delivery with automated quality gates and continuous cost optimization.

Conclusion: The Peer Reality Check

The challenge facing engineering leaders today is to ensure their initial AI-assisted coding gains translate into faster, higher-quality software delivery at scale. Organizations that remain focused on AI tools alone will continue to encounter the same structural bottlenecks. Those that invest equally in governance, context, trust, measurement, and cost optimization will be better positioned to move beyond the 10-15% ceiling and accelerate software delivery.

PAVE provides a structured path for making that transition. By treating AI as a program capability rather than an individual productivity tool, organizations can progressively increase AI autonomy while maintaining quality, controlling costs, and delivering measurable improvements across the SDLC.

The next phase of AI in software engineering will be defined by the operating models they build around it.

About Aspire Systems

Aspire Systems is a global technology services firm serving as a trusted technology partner for our customers. We work with some of the world's most innovative enterprises and independent software vendors, helping them leverage technology and outsourcing in our specific areas of expertise. Our core philosophy of "Attention. Always." communicates our belief in lavishing care and attention on our customer and employees.

For more info contact:
info@aspiresys.com or visit www.aspiresys.com