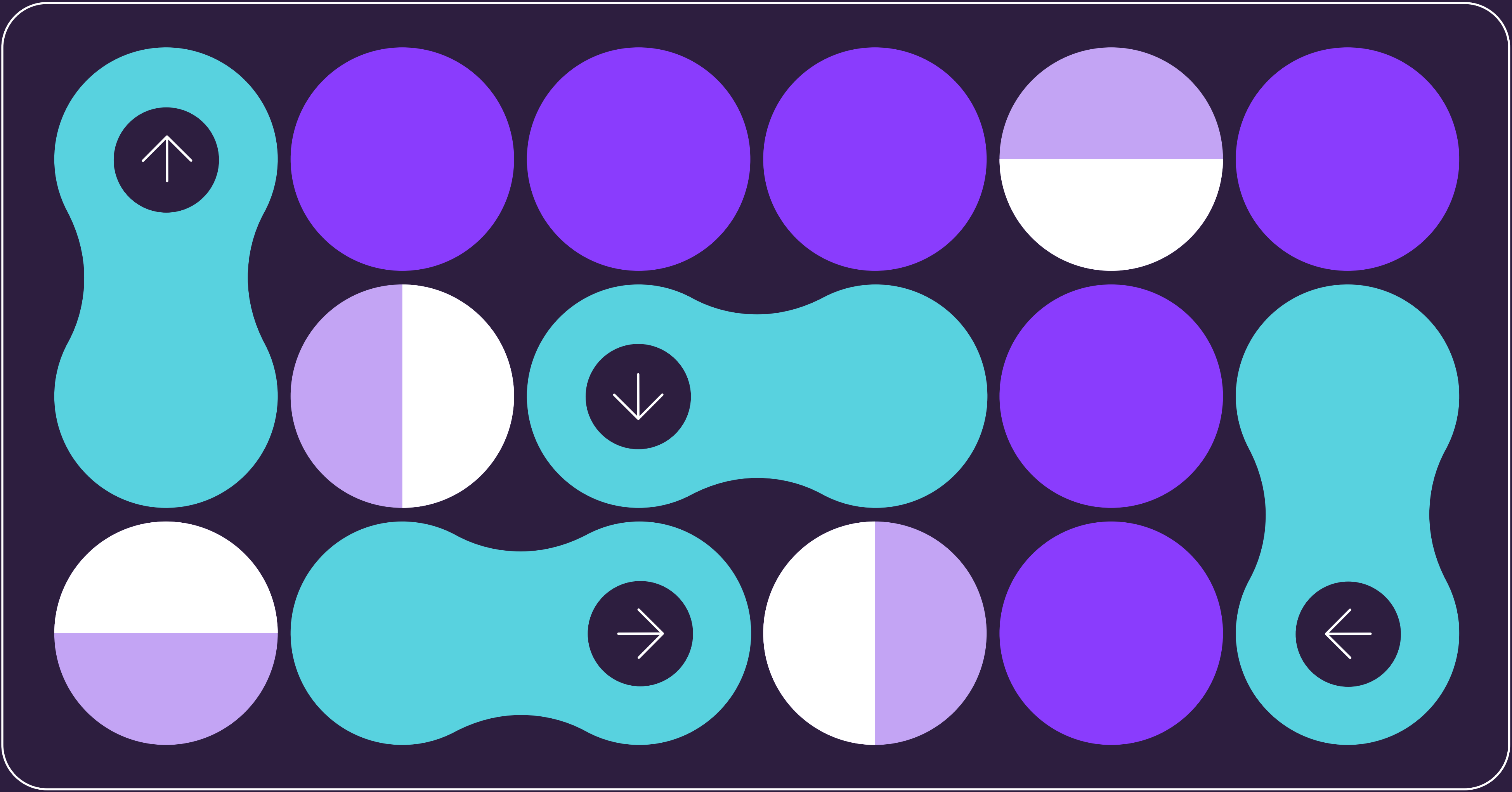
SOFTWARE ENGINEERING BENCHMARKS REPORT

‘26 | The AI productivity edition



LinearB

CREATED FROM A STUDY OF 8.1+ M PRS FROM 4,800 ENGINEERING TEAMS ACROSS 42 COUNTRIES.

Table of contents

Introduction

We're thrilled to be publishing the fifth annual edition of our Software Engineering Benchmarks Report. The 2026 report is the most comprehensive analysis of its kind, built from over 8.1 million pull requests across 4,800 development teams in 42 countries. This year, our research goes beyond traditional software delivery metrics to explore one of the most important questions facing engineering leaders today: **How is AI impacting software development?**

DATA SOURCED FROM

4,800 teams

8.1+ million pull requests

42 countries

Artificial intelligence has rapidly become embedded in the tools and workflows that engineers rely on every day. AI-assisted coding has increased output across the industry, but it has also exposed new challenges downstream: in review, testing, release, and governance. The question facing many leaders is no longer whether AI improves productivity, but how to measure and sustain that improvement across complex systems of people, tools, and processes.

At LinearB, we view AI Productivity as the next evolution of Software Engineering Intelligence. Measuring traditional outputs such as Cycle Time or PR Throughput remains essential, but leaders must now also understand how AI influences these metrics, and how it reshapes the systems, processes, and behaviors that underpin them. Assessing AI productivity now requires both quantitative and qualitative insight – capturing not just what's changing in performance, but how teams are experiencing and adapting to those changes.

That's why for 2026, we have introduced a new qualitative dimension to complement our quantitative data. In addition to the metrics pulled from across the software development lifecycle, this year's report includes insights from our **2026 AI in Engineering Leadership Survey**, a study capturing the perspectives of engineering executives, platform leaders, and DevEx professionals on the organizational impact of AI. Their experiences and observations bring the data to life, capturing real-world perspectives from the development community on how AI is transforming productivity, code quality, and DevEx.

We hope that you enjoy these fascinating new insights into how AI is reshaping the craft and culture of software engineering. Above all, the 2026 Software Engineering Benchmarks Report is designed to help you understand not only where your teams stand, but why – offering both data and perspective to guide smarter decisions in the AI era.

[illegible]

Yishai Beeri [CTO, LINEARB]

[IN THIS REPORT]

You'll find all metrics organized by the following criteria: 1) All Organizations, 2) by Size band, 3) by Geography. It's important to note that all data has been anonymized and normalized. For aggregation, we used the P75 (75th percentile) calculation. P75 is less sensitive to extreme values or outliers in the data, providing a robust and reliable measure.

[TOOLTIP]

WE'VE ORGANIZED THE DATA INTO THE FOLLOWING LEVELS OF PERFORMANCE FOR EACH METRIC:

🟩	**Elite** Top 10% the included Organizations
🟦	**Good** Top 30% the included Organizations
🟨	**Fair** Top 60% the included Organizations
🟥	**Needs focus** Bottom 40% the included Organizations

Software engineering benchmarks

BENCHMARKS BY ORG | 8,109,244 PULL REQUESTS | 4,813 TEAMS | 163,820 ACTIVE CONTRIBUTORS

Category	Metric	Elite	Good	Fair	Needs focus
Delivery	Coding Time (hours)	< 54 mins	54 mins - 4 hours	5 - 23	> 23
	Pickup Time (hours)	< 1	1 - 4	5 - 16	> 16
	Approve Time (hours)	< 10	10 - 22	23 - 42	> 42
	Merge Time (hours)	< 1	1 - 3	4 - 16	> 16
	Review Time (hours)	< 3	3 - 14	15 - 24	> 24
	Deploy Time (hours)	< 16	16 - 106	107 - 277	> 277
	Cycle Time (hours)	< 25	25 - 72	73 - 161	> 161
	Merge Frequency (per dev/week)	> 2.0	2 - 1.2	1.2 - 0.66	< 0.66
	Deploy Frequency (per service)	> 1.2	1.2 - 0.5	0.5 - 0.2	< 0.2
	PR Size (code changes)	< 100	100 - 155	156 - 228	> 228
	PR Maturity (%)	> 89%	89 - 83%	82 - 77%	< 77%
Predictability	Change Failure Rate (%)	< 1%	1 - 4%	5 - 17%	> 17%
	Refactor Rate (%)	< 11%	11 - 16%	17 - 22%	> 22%
	Rework Rate (%)	< 3%	3 - 5%	6 - 8%	> 8%
	Capacity Accuracy (%)	85 - 115%	75 - 85% or 115 - 125%	70 - 75% or 125 - 130%	> 70% or > 130%
	Planning Accuracy (%)	> 82%	82% - 64%	63% - 47%	< 47%
Project Management	Issues Linked to Parents (%)	> 90%	90 - 67%	66 - 56%	< 56%
	Branches Linked to Issues (%)	> 77%	77 - 62%	61 - 41%	< 41%
	In Progress Issues with Estimation (%)	> 55%	55 - 26%	25 - 14%	< 14%
	In Progress Issues with Assignees (%)	> 96%	96 - 84%	83 - 76%	< 76%

ENTERPRISE [1000+ EMPLOYEES]

Software engineering benchmarks

Category	Metric	Elite	Good	Fair	Needs focus
Delivery	Coding Time (hours)	< 2	2 - 6	7 - 26	> 26
	Pickup Time (hours)	< 2	2 - 4	5 - 13	> 13
	Approve Time (hours)	< 15	15 - 23	24 - 45	> 45
	Merge Time (hours)	< 2	2 - 4	5 - 16	> 16
	Review Time (hours)	< 3	3 - 13	14 - 23	> 23
	Deploy Time (hours)	< 25	25 - 173	174 - 347	> 347
	Cycle Time (hours)	< 27	27 - 89	90 - 165	> 165
	Merge Frequency (per dev/week)	> 1.4	1.4 - 0.9	0.8 - 0.5	< 0.5
	Deploy Frequency (per service)	> 0.7	0.7 - 0.3	0.2 - 0.1	< 0.1
	PR Size (code changes)	< 100	100 - 141	142 - 220	> 220
	PR Maturity (%)	> 89%	89 - 85%	84 - 78%	< 78%
Predictability	Change Failure Rate (%)	< 1%	1 - 2%	3 - 26%	> 26%
	Refactor Rate (%)	< 11%	11 - 17%	18 - 23%	> 23%
	Rework Rate (%)	< 3%	3 - 5%	6 - 8%	> 8%
	Capacity Accuracy (%)	85 - 115%	75 - 85% or 115 - 125%	70 - 75% or 125 - 130%	> 70% or > 130%
	Planning Accuracy (%)	> 82%	82% - 64%	63% - 47%	< 47%

BENCHMARKS BY ORG | 8,109,244 PULL REQUESTS | 4,813 TEAMS | 163,820 ACTIVE CONTRIBUTORS

SCALE-UP [200 - 1000 EMPLOYEES]

Software engineering benchmarks

Category	Metric	Elite	Good	Fair	Needs focus
Delivery	Coding Time (hours)	< 1	1 - 6	7 - 31	> 31
	Pickup Time (hours)	< 2	2 - 6	7 - 16	> 16
	Approve Time (hours)	< 17	17 - 24	25 - 46	> 46
	Merge Time (hours)	< 2	2 - 5	6 - 19	> 19
	Review Time (hours)	< 5	5 - 18	19 - 26	> 26
	Deploy Time (hours)	< 17	17 - 124	125 - 296	> 296
	Cycle Time (hours)	< 45	45 - 95	96 - 169	> 169
	Merge Frequency (per dev/week)	> 1.6	1.6 - 1.1	1 - 0.6	< 0.6
	Deploy Frequency (per service)	> 0.9	0.9 - 0.5	0.4 - 0.2	< 0.2
	PR Size (code changes)	< 103	103 - 157	158 - 212	> 212
	PR Maturity (%)	> 88%	88 - 82%	81 - 76%	< 76%
Predictability	Change Failure Rate (%)	< 1%	1 - 3%	4 - 13%	> 13%
	Refactor Rate (%)	< 12%	12 - 16%	17 - 22%	> 22%
	Rework Rate (%)	< 4%	4 - 5%	6 - 7%	> 7%
	Capacity Accuracy (%)	85 - 115%	75 - 85% or 115 - 125%	70 - 75% or 125 - 130%	> 70% or > 130%
	Planning Accuracy (%)	> 82%	82% - 64%	63% - 47%	< 47%

BENCHMARKS BY ORG | 8,109,244 PULL REQUESTS | 4,813 TEAMS | 163,820 ACTIVE CONTRIBUTORS

START-UP [0 - 200 EMPLOYEES]

Software engineering benchmarks

Category	Metric	Elite	Good	Fair	Needs focus
Delivery	Coding Time (hours)	< 1	1 - 4	5 - 23	> 23
	Pickup Time (hours)	< 2	2 - 5	6 - 17	> 17
	Approve Time (hours)	< 8	8 - 22	23 - 41	> 41
	Merge Time (hours)	< 2	2 - 3	4 - 17	> 17
	Review Time (hours)	< 3	3 - 13	14 - 24	> 24
	Deploy Time (hours)	< 15	15 - 94	95 - 268	> 268
	Cycle Time (hours)	< 25	25 - 71	72 - 164	> 164
	Merge Frequency (per dev/week)	> 2.2	2.2 - 1.4	1.3 - 0.8	< 0.8
	Deploy Frequency (per service)	> 1.4	1.4 - 0.6	0.5 - 0.3	< 0.3
	PR Size (code changes)	< 105	105 - 158	159 - 235	> 235
	PR Maturity (%)	> 89%	89 - 83%	82 - 77%	< 77%
Predictability	Change Failure Rate (%)	1%	1 - 4%	5 - 16%	> 16%
	Refactor Rate (%)	< 11%	11 - 16%	17 - 21%	> 21%
	Rework Rate (%)	< 3%	3 - 5%	6 - 7%	> 7%
	Capacity Accuracy (%)	85 - 115%	75 - 85% or 115 - 125%	70 - 75% or 125 - 130%	> 70% or > 130%
	Planning Accuracy (%)	> 82%	82% - 64%	63% - 47%	< 47%

BENCHMARKS BY ORG | 8,109,244 PULL REQUESTS | 4,813 TEAMS | 163,820 ACTIVE CONTRIBUTORS

Metrics definitions

DELIVERY

Coding Time

The time it takes from the first commit until a pull request is published. Short Coding Time correlates to low WIP, small PR Size, and clear requirements.

Pickup Time

The time a pull request waits for someone to start reviewing it. Low Pickup Time represents strong teamwork and a healthy review process.

Approve Time

The time from first comment to the first approval. This metric, along with Merge Time, is a more granular segment of Review Time.

Merge Time

The time from the first approval to merge. This metric, along with Approve Time, is a more granular segment of Review Time.

Review Time

The time it takes to complete a code review and get a pull request merged. Low Review Time represents strong teamwork and a healthy review process.

Deploy Time

The time from when a branch is merged to when the code is released. Low Deploy Time correlates to high Deploy Frequency.

Cycle Time

The time it takes for a single engineering task to go through the different phases of the delivery process from ‘code’ to ‘production’.

Merge Frequency

The total number of pull requests or merge requests merged by a team over a period of time.

Deploy Frequency

A measurement of how often code is released. Elite Deploy Frequency represents a stable and healthy continuous delivery pipeline.

PR Size

The number of code lines modified in a pull request. Smaller pull requests are easier to review, safer to merge and correlate to a lower Cycle Time.

PR Maturity

The ratio between the total changes added to a PR branch after the PR was published and the total changes in the PR.

Metrics definitions

PREDICTABILITY

Change Failure Rate (CFR)

The percentage of deploys causing a failure in production.

Rework Rate

The amount of changes made to code that is less than 21 days old. High Rework Rates signal code churn and is a leading indicator of quality issues.

Refactor Rate

Refactored work represents changes to legacy code. LinearB considers code “legacy” if it has been in your codebase for over 21 days.

Planning Accuracy

The ratio of planned work vs. what is actually delivered during a sprint or iteration. High Planning Accuracy signals a high level of predictability and stable execution.

Capacity Accuracy

Measures all completed (planned and unplanned) work as a ratio of planned work.

PROJECT MANAGEMENT

Issues Linked to Parents

The percentage of issues or tickets within your PM instance that are linked to a parent issue, such as an epic or story. This does not include subtasks.

Branches Linked to Issues

The percentage of code branches that contain a reference to specific PM issues, providing visibility into the alignment of code changes with planned tasks.

In Progress Issues with Estimation

The proportion of ongoing PM tasks that have time or effort estimates assigned.

In Progress Issues with Assignees

The percentage of active PM tasks that have a designated team member responsible for completing them.

Metrics definitions

AI

Agentic AI PRs

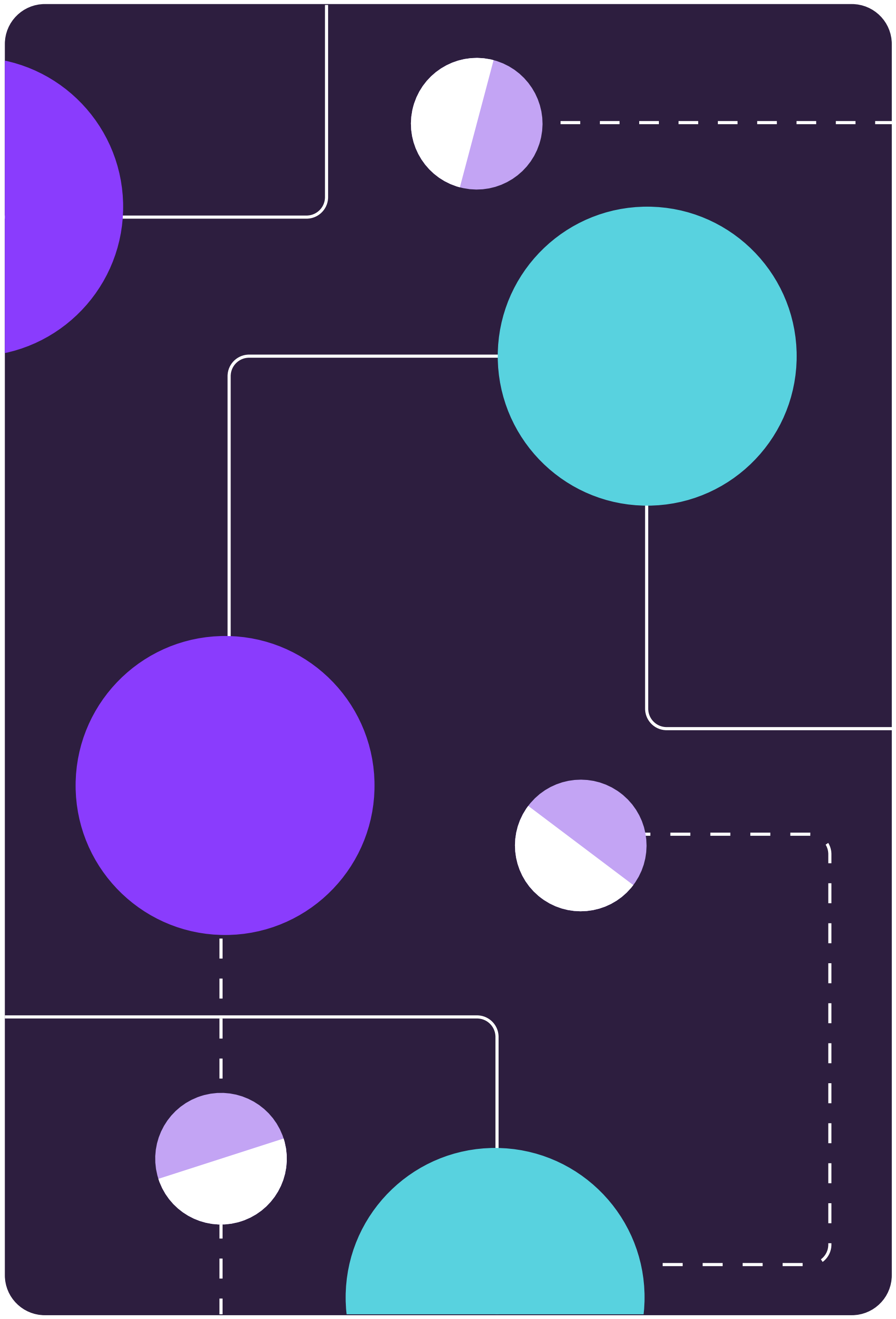
Pull requests created by AI agents, running advanced agentic flows (e.g. agents that are given a high-level task), and perform a full process for generating the code, committing it, and creating a pull request to get it merged. Sometimes these PRs can represent a smaller task such as fixing a simple bug, typo etc.

Examples of such agents include Devin, Copilot Coding Agent, OpenAI Codex, and others.

Acceptance Rate

The percentage of PRs that get merged within X=30 days of being created (or moved out of draft).

Note: We have standardized on a merge threshold of 30 days, but our insights hold for thresholds of 7d, 14d, and even no time limit for merge.



New this year: Acceptance Rate benchmarks

This year, we're excited to add a new benchmark to our study:

Acceptance Rate

Acceptance Rate is the percentage of PRs that get merged within X=30 days of being created (or moved out of draft).

The Acceptance Rate metric and benchmark are particularly relevant to AI-generated code, where ownership patterns and acceptance flows are being challenged. And indeed, early Acceptance Rate benchmarks reveal that AI-generated PRs operate in a completely different performance range than manual ones, underscoring how new and uneven this workflow still is.

ACCEPTANCE RATE BENCHMARKS	ELITE	GOOD	FAIR	NEEDS FOCUS
All PRs	> 95%	91 - 95%	85 - 90%	< 85%
Manual PRs	> 95%	92 - 95%	87 - 91%	< 87%
Agentic AI PRs	> 71%	61 - 71%	42 - 60%	< 42%

While teams qualify as Elite only when they exceed a 95% Acceptance Rate for manual PRs, AI PRs reach the same tier at just above 71%, a dramatically lower bar that reflects the current reality of how teams handle AI contributions.

FOR AI PRS, MOST TEAMS FIND IT HARD TO EXCEED ACCEPTANCE RATE OF **60%**

This divide continues across all benchmark tiers, with most teams finding it hard to exceed an Acceptance Rate of 60% for AI PRs – far lower than observed for manual PRs. This hints at the fundamental challenge with AI in the SDLC: creating more code faster does not translate to higher delivery. Reviewers approach AI work with greater scrutiny (see qualitative insights below); AI output still lacks the consistency and trust levels of manual code; and unclear ownership leaves many PRs abandoned. As a result, AI PRs are not just underperforming relative to manual PRs; they have created a new benchmark range that shows more code does not necessarily result in increased delivery.

QUALITATIVE INSIGHTS AI IN LEADERSHIP SURVEY

What engineering leaders are saying

Our survey responses make it clear that context (not just correctness) is the single biggest barrier to merging AI-generated PRs. Leaders told us that AI often produces changes that look reasonable on the surface but lack the deeper rationale, intent, or situational awareness reviewers need to feel confident approving them. Without that context, even clean-looking code can feel risky: reviewers worry about hidden side effects, missing assumptions, or logic that doesn't align with how the system actually works. As a result, they slow down, ask more questions, or reject the PR entirely – not because the code is wrong, but because they can't trust what they can't fully understand.

Here's how they put it:

"AI generates advanced code the author could not explain."

"Context is often wrong or missing."

"Some of the code AI generates is very large and may not [be] needed for the context..."

"Without context it generates more work than a junior developer. The workload increases for the reviewer."

AI productivity insights

[DISCLAIMER]

IT'S IMPORTANT TO NOTE THAT CORRELATION DOES NOT INDICATE CAUSATION. HOWEVER, THE INSIGHTS THIS DATA EXPOSES ALIGN CLOSELY WITH THE QUALITATIVE AND ANECDOTAL RESEARCH WE'VE GATHERED FROM LINEARB USERS OVER THE PAST YEAR.

AI code in the SDLC

[INSIGHT 01]	AI-Assisted PRs are 2.6x larger than Unassisted ones.
[INSIGHT 02]	AI PRs have a PR Pickup Time 5.3x longer than Unassisted ones.
[INSIGHT 03]	Acceptance (Merge) rate for AI PRs is less than half that of manual PRs.
[INSIGHT 04]	Heavier Agentic AI adoption is not translating to greater value delivery.
[INSIGHT 05]	AI PR 30-day Acceptance (Merge) Rate varies widely by tool.
[INSIGHT 06]	AI use is biased towards new code, rather than refactoring.

AI Code in the SDLC

In this section, we'll examine benchmarks across three types of pull requests: Agentic AI PRs, AI-Assisted PRs, and Unassisted PRs (as defined below).

[PULL REQUEST TYPES]

Agentic AI PRs

We define Agentic AI PRs as pull requests created by AI agents, such as Devin, Copilot Coding Agent, OpenAI Codex, and others. They represent advanced agentic flows, typically agents that are given a task (directly, or via a Jira ticket, for example), and perform a full process for generating the code, committing it and creating a pull request to get it merged. Sometimes these PRs can represent a smaller task such as fixing a simple bug, typo etc.

Earlier examples of (non-AI) bot PRs include dependency management PRs created by Dependabot, Renovate and Snyk, for example, to bump versions of dependency libs to patch security issues.

AI-Assisted PRs

Defined as a code change authored or significantly shaped with the help of AI-powered development tools. These PRs originate from human intent and ownership, with the AI contributing suggestions, code blocks, or refactors that the developer integrates into the final change.

AI-assisted PRs differ from agentic flows in that the AI does not independently interpret tasks, commit code, or open the PR. Instead, the developer remains in control of the workflow, using AI as an augmenting tool rather than an autonomous author.

Unassisted PRs

Pull requests authored entirely by developers without substantial use of AI tools for code generation, modification, or refactoring.

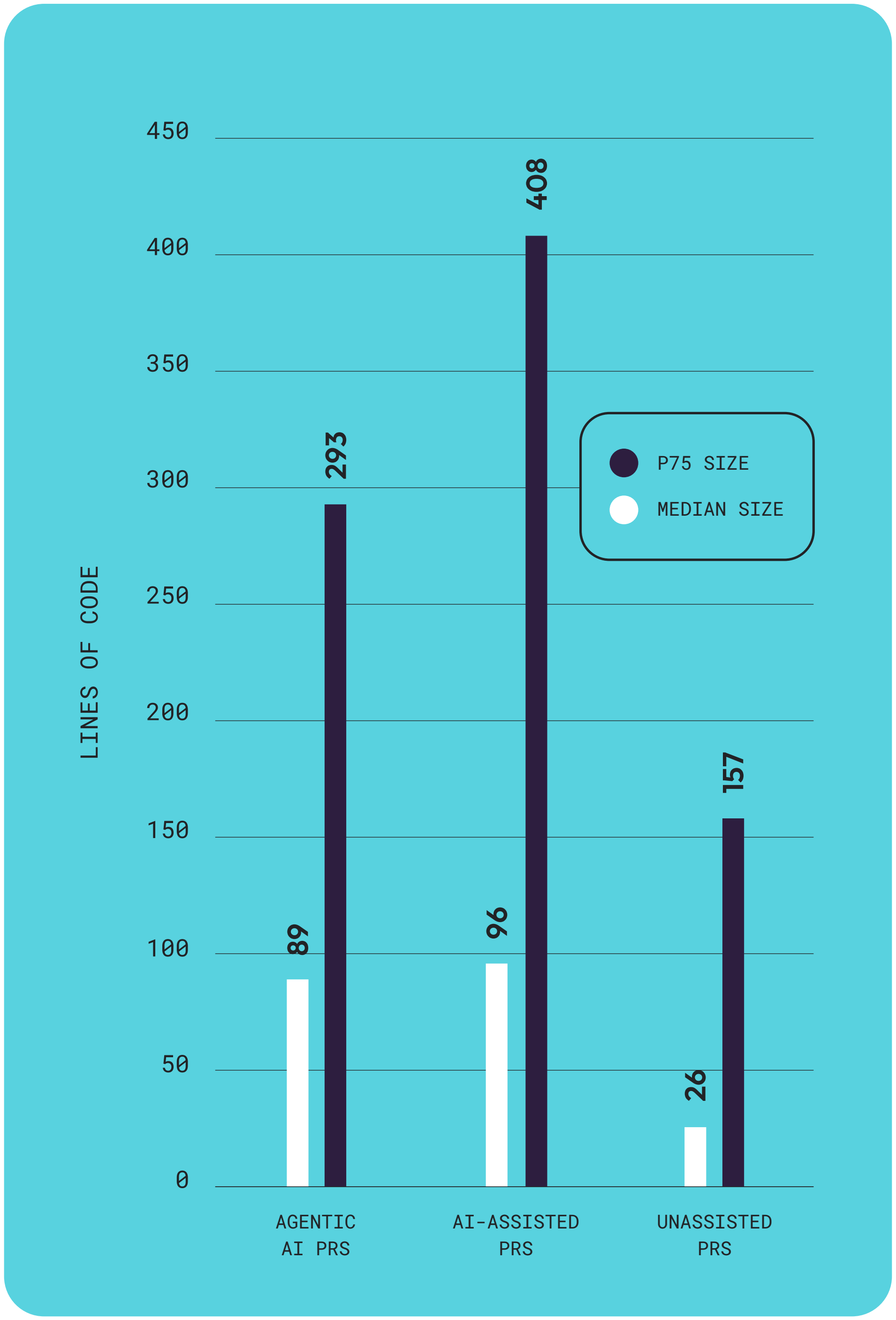
As agentic AI systems take on more of the end-to-end development flow (interpreting tasks, generating code, and opening PRs without human authorship, for e.g.), their impact on engineering workflows becomes increasingly visible. While adoption is surging and AI has become a daily tool for most developers, the technology itself is still early in its maturity, and that shows in the PRs it produces. PRs with AI involvement (both Agentic AI PRs and AI-Assisted PRs) often arrive larger, broader in scope, and less predictable than manual changes, placing new demands on reviewers who must interpret intent and assess risk without the benefit of human context. The result is a growing mismatch between widespread AI usage and the workflows built to support it: AI helps developers write code faster, but reviewers are being asked to shoulder more complexity with tools that are still evolving. At this moment, AI is both accelerating code output and making the review process more complex, highlighting just how new the technology still is.

[INSIGHT 01]

AI-Assisted PRs are 2.6x larger than Unassisted ones.

	MEDIAN SIZE	P75 SIZE
Agentic AI PRs	89	293
AI-Assisted PRs	96	408
Unassisted PRs	26	157

AI-generated PRs tend to come in much bigger than the ones written by humans, and that alone changes the review experience. At the p75 mark, Agentic AI PRs reach 293 lines of code compared to 157 lines for Unassisted PRs, which means reviewers suddenly have much more to process. Larger diffs often touch more files and more parts of the system, raising the odds of downstream issues and creating higher levels of code complexity. And because



there's just more code to understand, reviewers have to spend extra time figuring out how everything fits together before they can even think about suggesting changes. Additionally, AI-Assisted PRs are now the largest of all three groups, hitting 408 lines at P75, suggesting that when developers rely on AI to generate or refine code, the total volume of changes still grows quickly, even when the developer retains authorship and intent. All of this adds up to a review process that feels heavier from the start, even when the code itself might be perfectly valid.

Assisted vs. Unassisted commits: File and line count comparison

	ASSISTED COMMITS	UNASSISTED COMMITS
Average number of files modified	4.2	7.51
Average code lines	273	498
Average code lines per file	65	66

Diving inside the PRs and into the separate commits, we see a different pattern. AI-assisted commits tend to be more concentrated, touching fewer files on average (4.2, compared to 7.51 for non-assisted commits), while maintaining a similar average size per file (65 vs. 66 lines). This seems to indicate that AI tools, which create commits for developers, are more "disciplined" with regard to commit boundaries, whereas humans tend to accumulate more work into each commit. Taken together, these averages show that both Agentic AI and AI-Assisted PRs consistently introduce larger code changes than Unassisted PRs, amplifying the volume reviewers must evaluate while shifting the shape and distribution of the underlying work.

QUALITATIVE INSIGHTS

What engineering leaders are saying

When we asked engineering leaders, "What's been the biggest challenge or concern with using AI in your role?"*, a noticeable subset of responses pointed directly to the growing size and scope of AI-generated changes.

Here's how they put it:

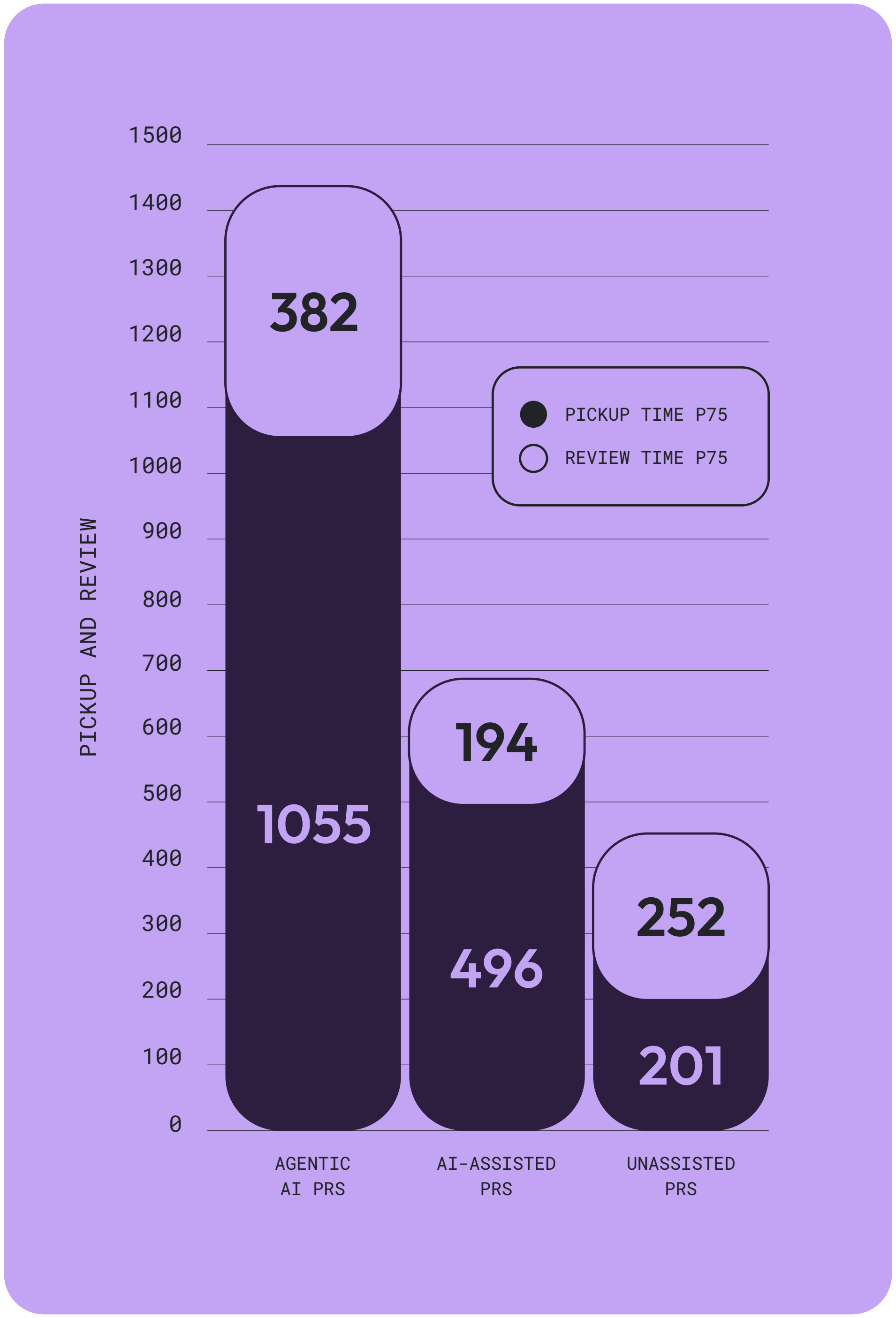
"Verbose code generation in distributed production systems."

> “AI tools tend to make changes larger than the request’s scope.”
>
> “Being able to fully trust the results. Getting WAY more text/information back than you wanted or needed, which slows you down because you have to hunt for what you actually were interested in (can somewhat be dealt with by writing better prompts, but it is still pretty common even with great prompts).”

[INSIGHT 02]

Agentic AI PRs have a PR Pickup Time 5.3x longer than Unassisted ones.

	PICKUP TIME P75	REVIEW TIME P75
Agentic AI PRs	1055	382
AI-Assisted PRs	496	194
Unassisted PRs	201	252



AI-generated PRs experience an imbalance in the review process: they wait significantly longer to be picked up, yet once someone starts reviewing them, the review itself is only somewhat slower compared to Unassisted PRs. At the P75 mark, Agentic AI PRs sit idle for 1,055 minutes before a reviewer even opens them, compared to just 201 minutes for Unassisted PRs. AI-Assisted PRs fall between the two, with a P75 Pickup Time of 496 minutes, suggesting that AI involvement introduces additional hesitation before reviewers engage, possibly due to larger diffs (see previous insight), unfamiliar code patterns, or a sense that "ownership" of the work is less clear.

Once a reviewer does begin, the dynamic is different. At the P75 mark, AI-assisted PRs are reviewed in just 194 minutes, faster than the 252-minute review time for Unassisted PRs. Agentic AI PRs are reviewed in 382 minutes, the slowest of the three. This behavior hints at two patterns: first, reviewers may be scanning for high-level issues rather than deeply evaluating the full changeset, especially when the code originates from or is heavily influenced by AI. Second, in AI-generated PRs, there is often no clear owner for the PR who will push to finalize the review and get the PR accepted. The combination of slow starts and fast finishes paints a picture of teams still finding their footing with AI-enabled work, unsure how to approach it at first, yet eager to move quickly once they do.

What engineering leaders are saying

Our survey responses mirrored this quantitative data almost exactly: once reviewers begin looking at an AI-generated PR, they tend to move quickly and focus on high-level issues – but getting to that point takes much longer. Leaders described AI-driven reviews as faster yet more shallow, with engineers often scanning for red flags rather than deeply understanding the change. At the same time, many expressed hesitation to even open an AI PR due to uncertainty, additional mental load, or concern that the code may contain subtle errors or lack critical context. Together, these patterns help explain why AI PRs sit idle far longer than manual ones, yet are reviewed in significantly less time once picked up.

When we asked, "Has AI changed how your team collaborates or reviews code? If so, how?", here's what they shared:

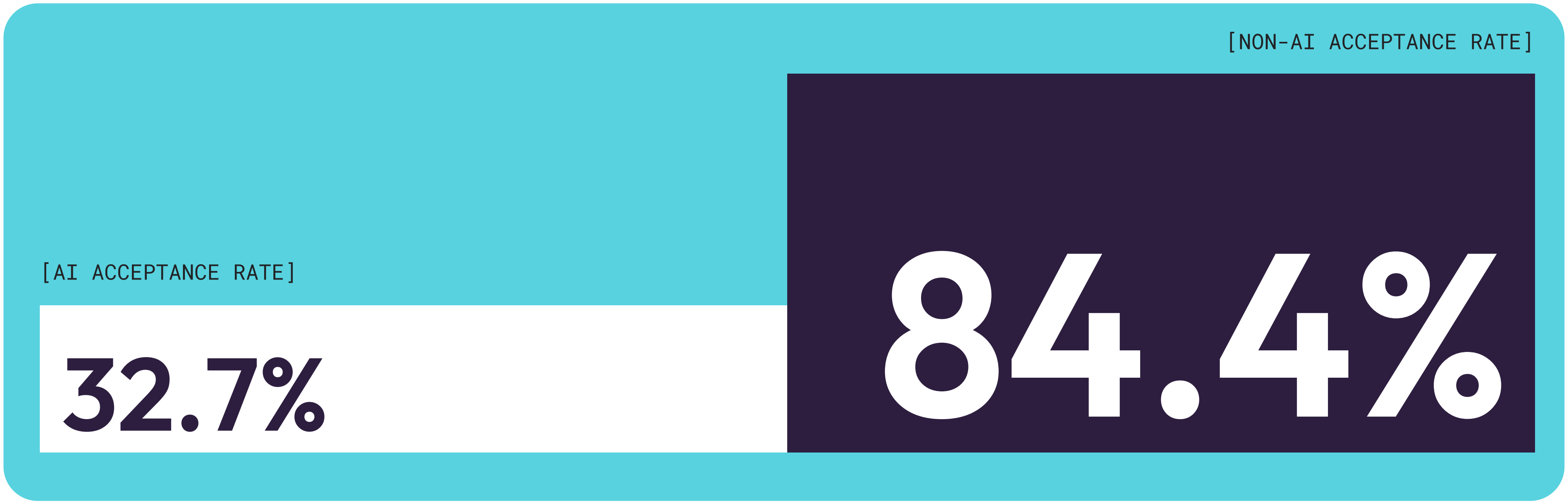
"A larger amount of code slips into production without proper review (rubber stamping)."

> “AI sometimes offers up mistaken suggestions and non-working code. Also, it can be too agreeable with no critical “thinking.”
>
> “It’s been challenging for us to be able to fully trust the results.”
>
> “AI does not always follow standards. It sometimes writes solutions that look correct, but are lacking in a vital way.”

[INSIGHT 03]

Acceptance (Merge) rate for AI PRs is less than half that of manual PRs.

With manual PRs merging at 84.4% and AI PRs merging at just 32.7%, teams are clearly treating AI-generated code differently. Some of this gap likely stems from unclear ownership: the AI PRs don’t typically have a natural human owner whose goal is to get them approved and merged. Without this push, they tend to get ignored and left to rot. Compounding this, in many cases, agentic AI flows are deployed to create PRs for low-priority work, catching up on small backlog items, etc. – work that commands less team focus to begin with. Finally, even when trying to promote the PRs, reviewers are more hesitant – as AI PRs tend to be larger and more complex, which naturally increases the chances of errors or edge cases that reviewers want to double-check. The result is a consistently lower Acceptance Rate that reflects the growing pains of integrating AI into existing development workflows.



What engineering leaders are saying

Trust remains one of the biggest hurdles for AI-generated PRs. Even when the code compiles or appears reasonable, many leaders struggle to rely on AI output with the same confidence they place in their teammates' work. Our survey respondents described AI-generated changes as unpredictable, inconsistent, or lacking the reliability required for a clean merge. This underlying skepticism naturally leads reviewers to hesitate, dig deeper, or reject the PR altogether, slowing down acceptance and widening the performance gap between AI and manual code.

In response to the question, "What's been the biggest challenge or concern with using AI in your role?", leaders shared:

"General distrust of AI code quality..."

"There is now lower quality code sent for review."

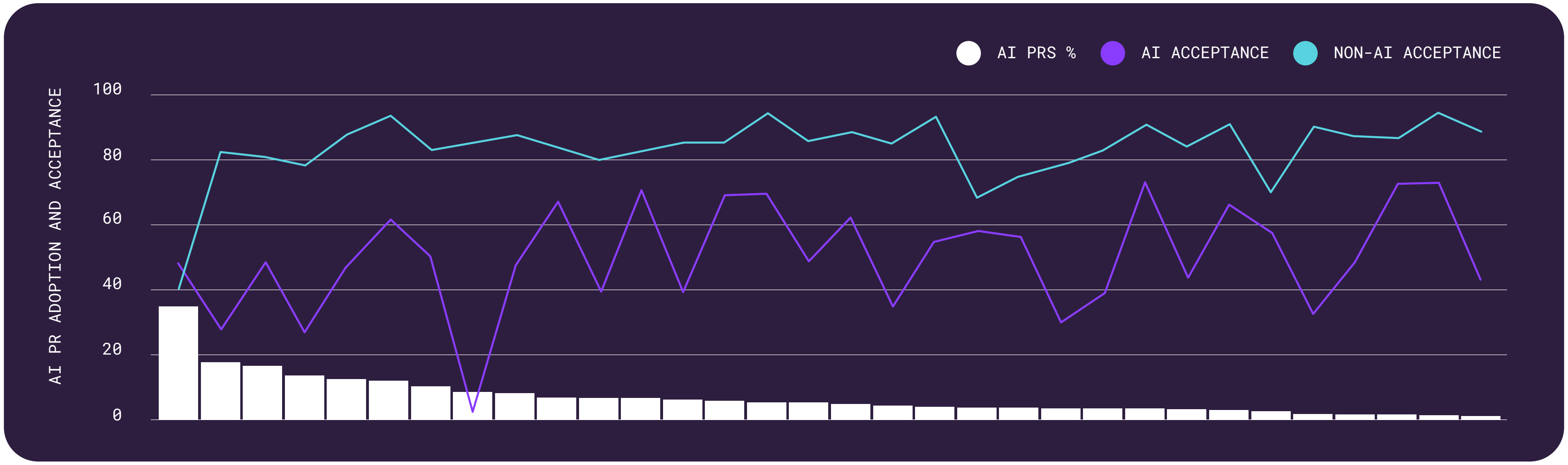
"AI output feels risky or unpredictable."

"The amount of slop it introduces."

"Trust in its output."

[INSIGHT 04]

Heavier Agentic AI adoption is not translating to greater value delivery.



Even among organizations that are further along in agentic AI adoption, with a larger portion of AI-driven PRs, Acceptance Rates don't show any meaningful improvement. In the data, orgs with higher AI PR adoption rarely outperform those with lower adoption.

In fact, most follow the same pattern of significantly lower acceptance for AI-generated work compared to manual PRs. This suggests that simply using AI more doesn't resolve the ownership challenges inherent in AI PRs and does not result in greater reviewer trust. Teams may still be experimenting with workflows, calibrating expectations, or figuring out how AI-generated code fits into their existing standards.

One notable outlier in our dataset (an org using Devin at very high adoption with acceptance near parity) shows that improvements are possible, but far from the norm. Overall, adoption alone isn't the driver. Process maturity, tooling, and reviewer confidence appear to matter far more than volume.

QUALITATIVE INSIGHTS AI IN LEADERSHIP SURVEY

What engineering leaders are saying

Even organizations using AI heavily still struggle with reliability and reproducibility – two qualities reviewers depend on when deciding whether to merge a PR. What works well for one developer might break down for another, and differences between models, tools, and workflows only widen that gap. These inconsistencies make it hard for teams to build trust in AI-generated work, even as usage grows rapidly.

To the question, "What's been the biggest challenge or concern with using AI in your role?", we also heard responses such as:

"What works well for one person is difficult to scale, also, I suspect our organization is likely to select tools based on sales reps words rather than pure quality of tooling."

"Reliability."

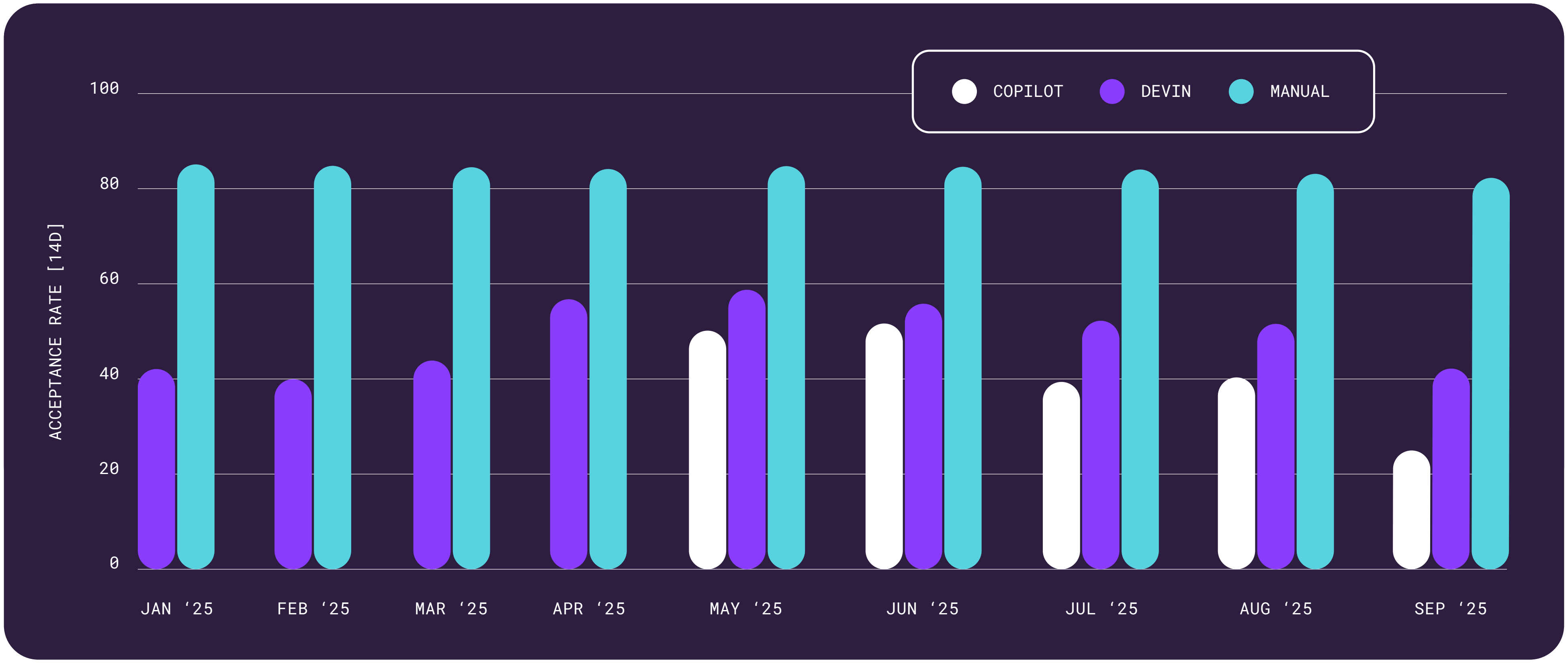
"Maintainability and consistency of quality."

"Standardising and not overusing."

These firsthand experiences reinforce a key point in the Acceptance Rate data: more AI PRs doesn't mean better AI PRs – and higher adoption doesn't equate to higher reviewer confidence or more successful merges.

[INSIGHT 05]

AI PR 30-day Acceptance (Merge) Rate varies widely by tool.



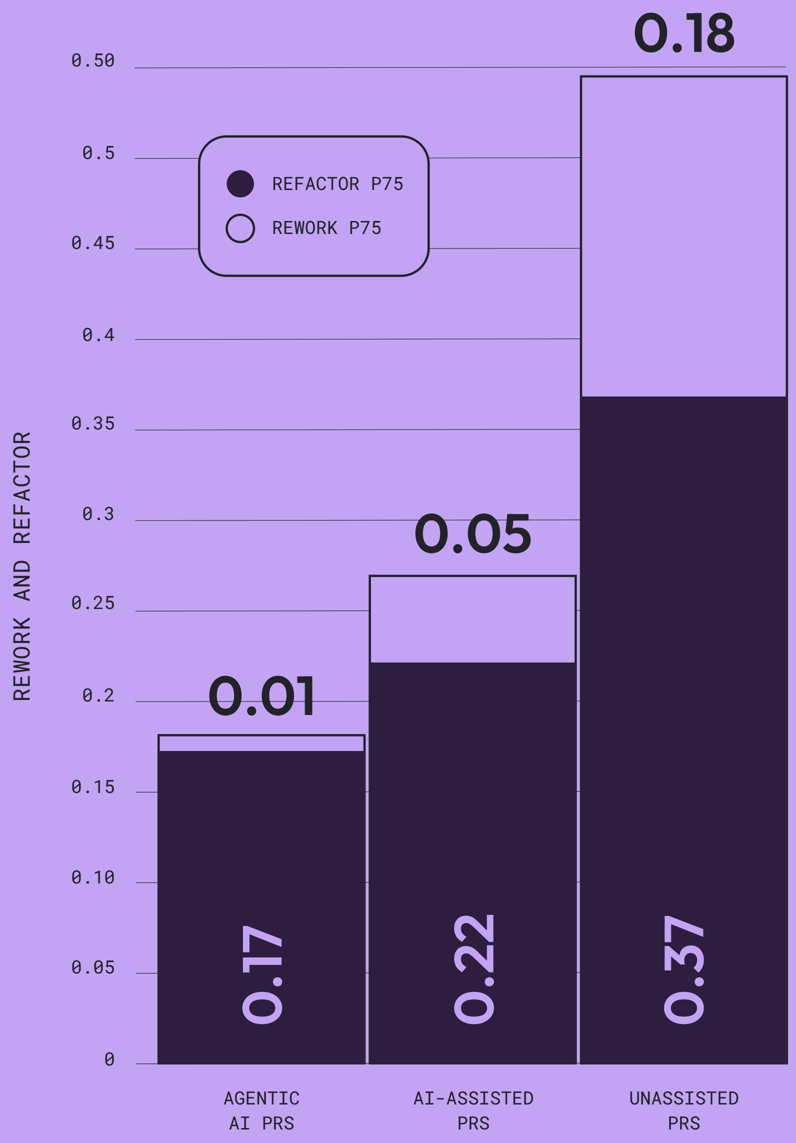
Not all AI coding agents perform the same when it comes to merging outcomes. Rather, the data shows a clear split: while manual PR acceptance stays steady around 80-82% month after month, bot performance fluctuates significantly depending on the tool. Devin demonstrates a noticeable improvement beginning in April, with its Acceptance Rates climbing closer to human baselines over time.

Copilot, on the other hand, shows a decline starting in May, suggesting either growing reviewer skepticism or quality issues emerging as usage scales. These differences highlight that "AI PRs" aren't a single category. Each tool brings its own coding patterns, error profiles, and trust dynamics. As a result, engineering teams may find themselves calibrating review expectations bot by bot, rather than treating all AI-generated code as equal.

[INSIGHT 06]

AI use is biased towards new code, rather than refactoring.

Our data shows a clear separation in how AI-generated, AI-assisted, and Unassisted PRs interact with existing code. At the P75 mark, Unassisted PRs show the highest rate of existing code modification, with a Refactor Rate of 0.37 – more than 1.5x higher than both Agentic (0.17) and AI-Assisted PRs (0.22). In contrast, Agentic AI PRs show almost no Rework activity, and AI-Assisted PRs exhibit only minimal levels. This pattern indicates that human-authored PRs remain far more likely to involve changes to recently written or legacy code, whereas AI-driven and AI-assisted PRs tend to introduce new code paths rather than adjust or correct existing ones. Perhaps the apparent ease of creating (a lot of) new code with AI is driving developers to focus on new code creation when turning to AI tools.



The DevEx of AI

[INSIGHT 01] AI adoption continues to rise, increasing from 71.6% in early 2024 to 88.3% in 2025.

[INSIGHT 02] Confidence in AI code quality is lukewarm.

[INSIGHT 03] 45% of orgs are not measuring AI impact, but 75% of managers attest to some gains.

The DevEx of AI

Developer Experience is increasingly shaped by AI adoption, with usage climbing steadily since 2025 (see Insight #1). However, confidence in AI-generated code quality remains lukewarm, and nearly 50% of organizations lack formal mechanisms to measure its impact, even as managers report perceived productivity gains. This gap highlights a core DevEx truth: teams can't deliver their best work when they don't have confidence in the tools they use. Without reliable AI outputs and clear evaluation frameworks, adoption alone can introduce friction, and ultimately reduce effective delivery. Integrating AI successfully isn't just about rolling out a new tool. Doing so responsibly requires trust between teams, open communication, and metrics that reveal how AI truly affects developer workflows.

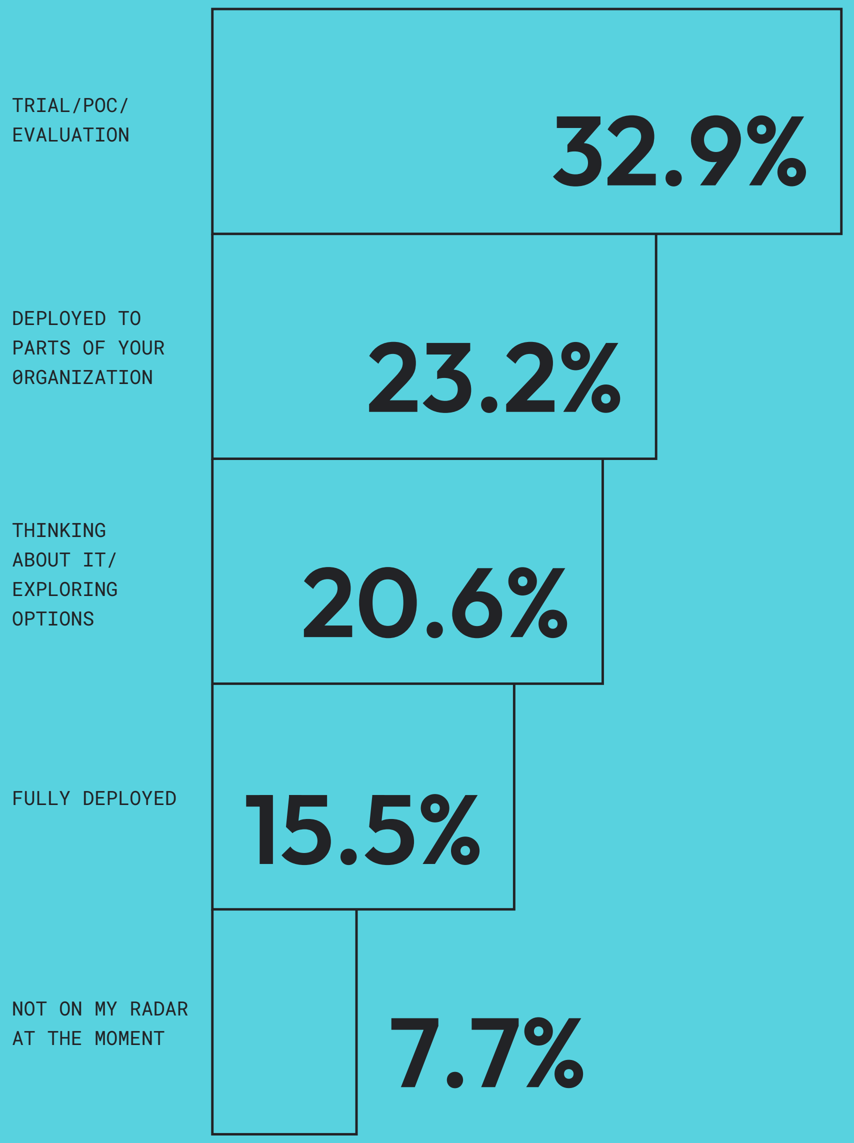
[INSIGHT 01]

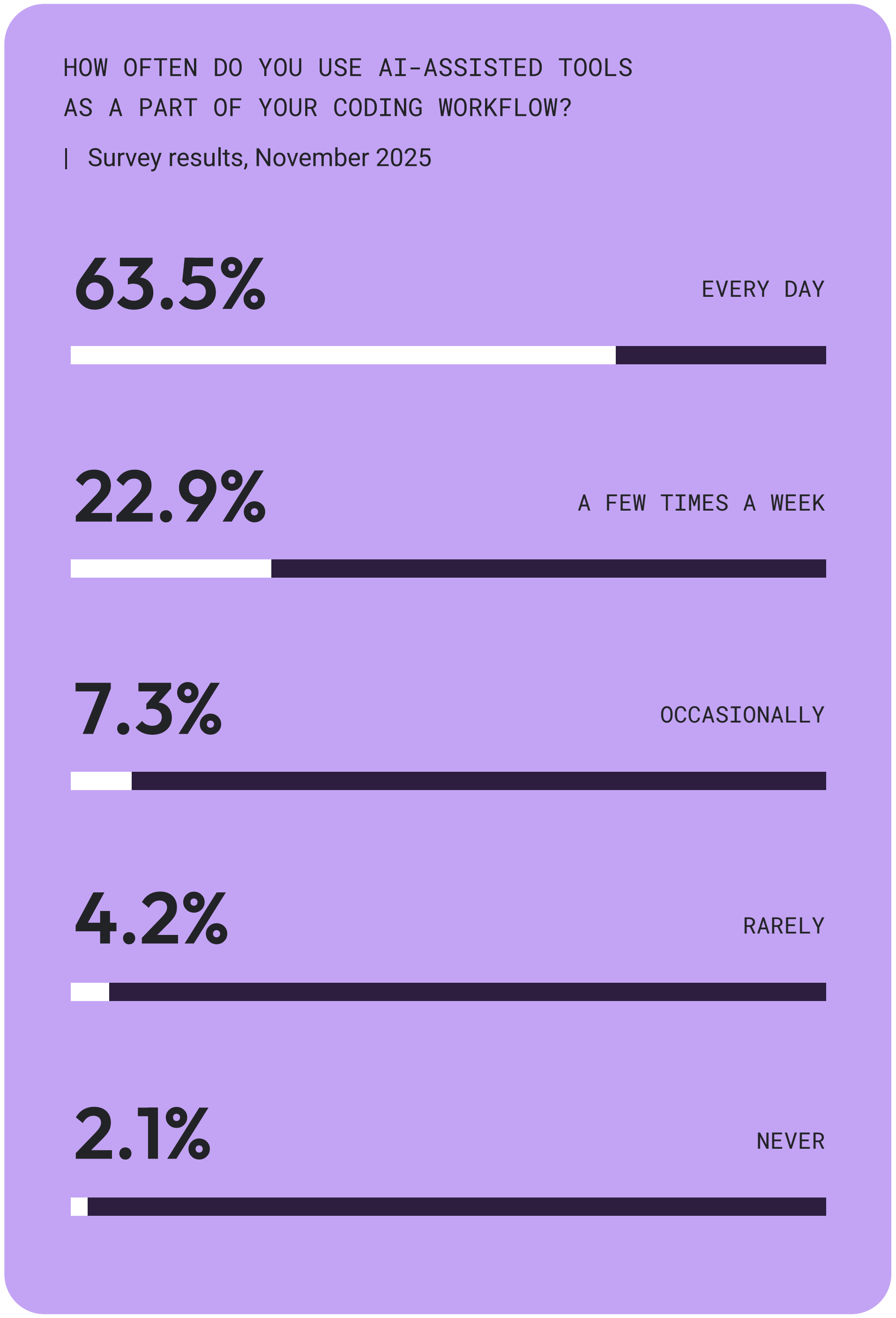
AI adoption continues to rise, increasing from 71.6% in early 2024 to 88.3% in 2025.

In January 2024, we conducted a Generative AI survey that gave us an early snapshot of how organizations were beginning to adopt AI tools.

DESCRIBE YOUR CURRENT STATUS IN REGARDS TO GENERATIVE AI CODING TOOL ADOPTION?

| Survey results, January 2024





At that time, 71.6% of organizations reported being either in or through a GenAI tool adoption process, spanning initial evaluation through full deployment. Our 2025 data reflects a 23% increase from that baseline, indicating that AI usage has advanced far beyond rollout phases and is now firmly embedded in day-to-day developer activity.

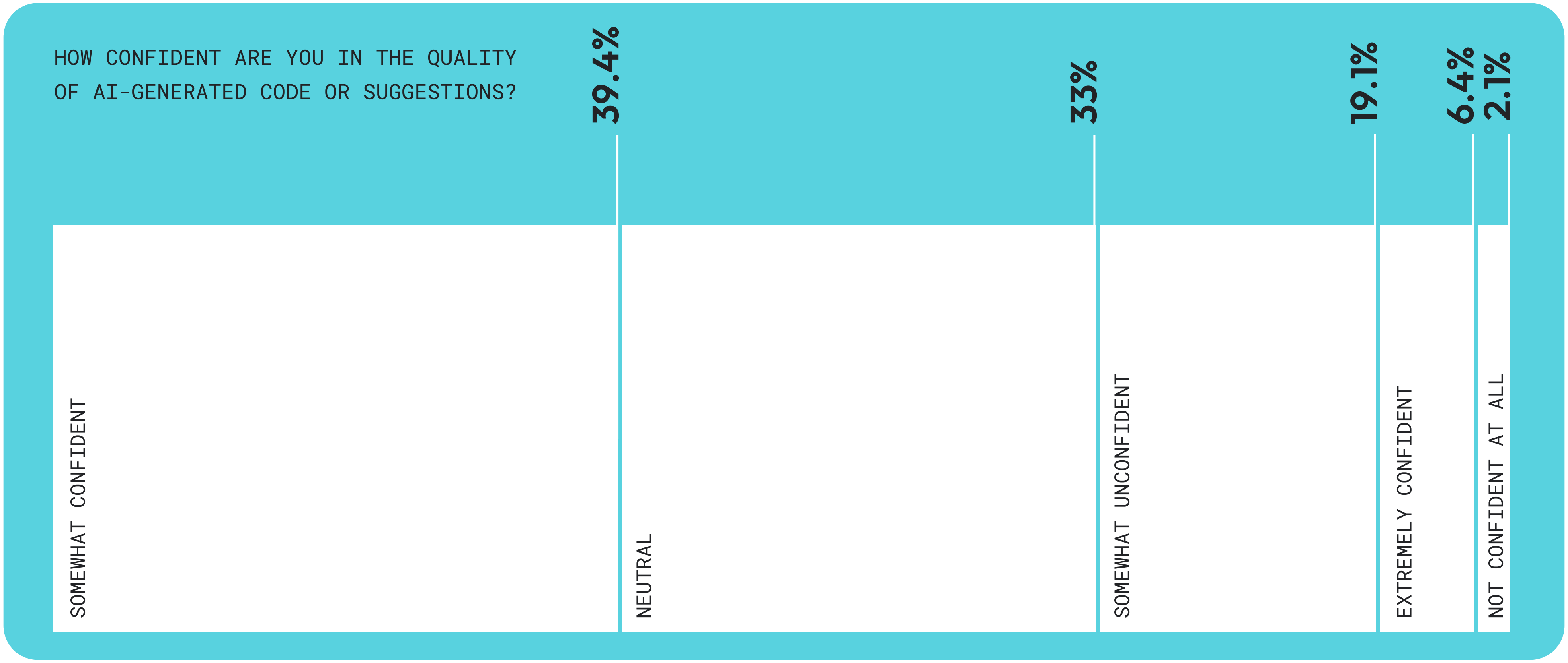
Our 2025 data revealed that nearly two-thirds of respondents (64.9%) are using AI-assisted tools every single day, and another 23.4% use them a few times a week, meaning close to 90% of developers are now relying on AI as a regular part of how they write and review code. Looking at lower-frequency usage, our data shows that only a small minority use AI occasionally (6.4%) or rarely (3.2%), and an even smaller fraction (just 2.1%) say they never use it at all. It's clear that AI is no longer a niche preference or an experimental fad. It's become a routine part of the modern development experience, shaping everything from how engineers ideate and debug to how they collaborate and deliver.

[INSIGHT 02]

Confidence in AI code quality is lukewarm.

Our AI in Engineering Leadership Survey revealed that confidence in AI-generated code follows a near perfect parabola – peaking at "somewhat confident" and tapering off evenly toward both high and low extremes. This shape is a strong signal of a technology that has achieved broad adoption, but still sits in a new and maturing market where trust hasn't fully solidified.

This qualitative data also helps explain the significantly lower Acceptance Rates we see in the data. While 39.4% of respondents say they are "somewhat confident," an almost equal share fall into neutral (33%) or somewhat unconfident (19.1%) categories – showing that most reviewers are hesitant. Only 6.4% report being "extremely confident," and a small but meaningful 2.1% say they are not confident at all. This confidence gap likely directly contributes to more scrutiny, slower review pickup, and ultimately fewer AI PRs getting merged, reinforcing that trust is a central barrier to AI adoption.

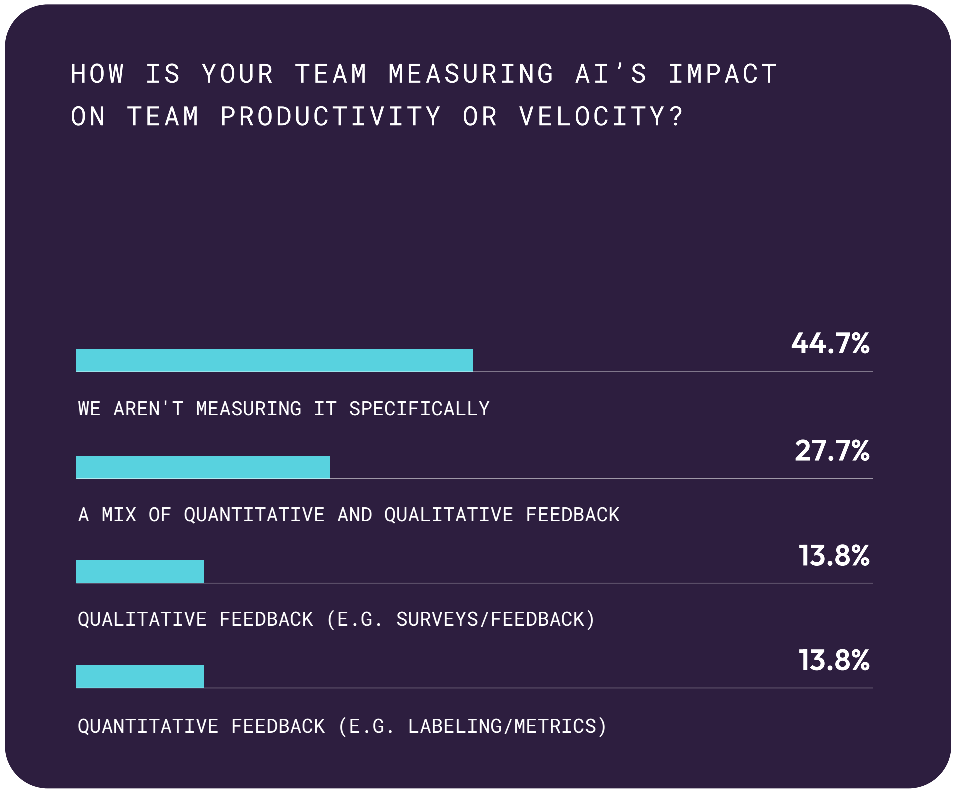


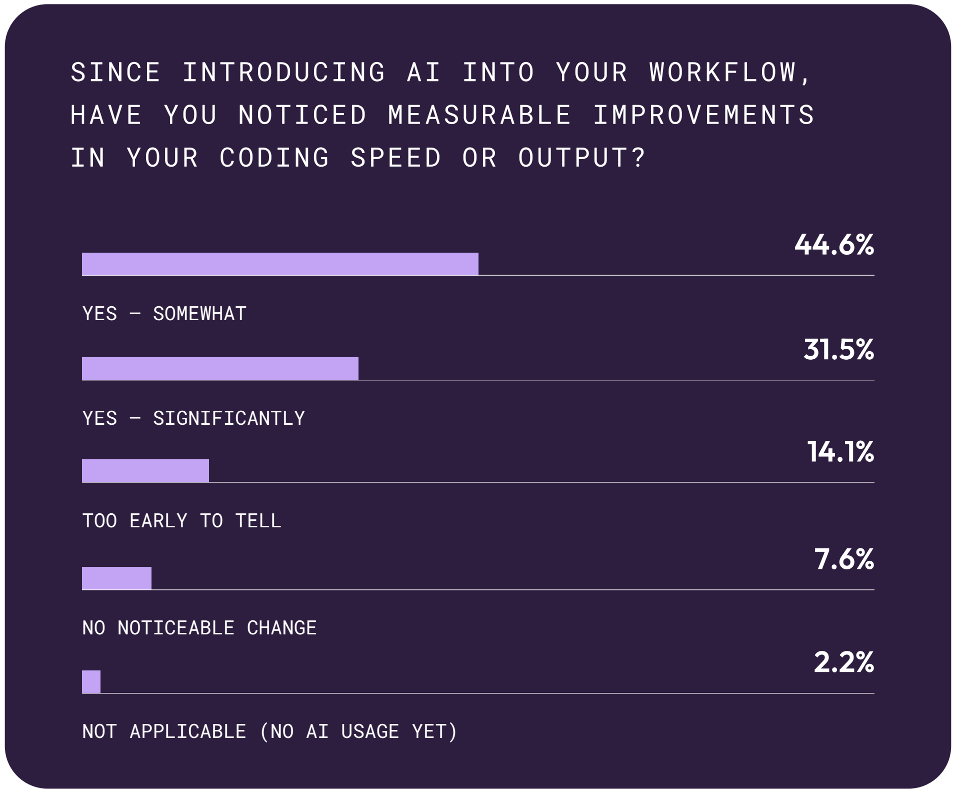
[INSIGHT 03]

45% of orgs are not measuring AI impact, but 75% of managers attest to some gains.

Survey results show that while a majority of engineering leaders (76.1%) report seeing measurable productivity improvements from AI – ranging from "somewhat" to "significant" – 44.7% of organizations are not currently measuring AI's impact in any formal way.

Only 27.7% use a mix of quantitative and qualitative data, and the remaining groups rely solely on qualitative feedback (13.8%) or quantitative metrics alone (13.8%). This indicates that most reported productivity gains are based on anecdotal observations like adoption rates, versus directly measured delivery impact. As AI becomes more integrated into development workflows, this gap between perceived impact and actual measurement highlights an emerging need for clearer instrumentation to track AI's contribution to team delivery.





LinearB

Built for modern software delivery

DevEx, Platform, and AI enablement teams now own the systems and workflows that transform raw code into reliable delivery. They need purpose-built tools to measure and manage AI's downstream impact across contributors and pipelines.

"We've improved our communication and the way we conduct our sprints. LinearB gave my team a common, data-backed language.

Alex L.

Senior Engineering Manager

Yum!

AI & Developer Productivity Insights

Track the impact of AI coding tools on delivery velocity, code quality, and team health. Combine operational metrics and surveys to get a complete performance view.

AI Code Reviews

Catch security risks, bugs, performance issues, and spec mismatches before code is merged. Enforce standards and compliance without slowing teams down.

DevOps Workflow Automation

Automate PR routing, approvals, and test enforcement with policy-based workflows. Eliminate manual review work and delivery bottlenecks.

Executive Reporting & ROI

Generate audit-ready reports that connect engineering costs — AI, people, projects and processes — to business results.

Developer Experience Optimization

Identify friction points with delivery metrics, qualitative feedback, and MCP-driven insights. Reduce context switching and improve team focus with AI-powered iteration retrospectives.

LinearB engineering productivity by the numbers

syngenta

81%

reduction in Cycle Time

AGRICULTURAL TECH

Yum!

321 hrs

automated per month

FAST-FOOD

duda

+20%

Deployment Frequency

SOFTWARE

Expedia

+22%

DSAT

E-COMMERCE

rabbitcare

51%

reduction in Cycle Time

FINANCIAL SERVICES

Edenred

$342k

in cost savings

FINANCIAL SERVICES

Super.com

11%

increase in
Planning Accuracy

TRAVEL

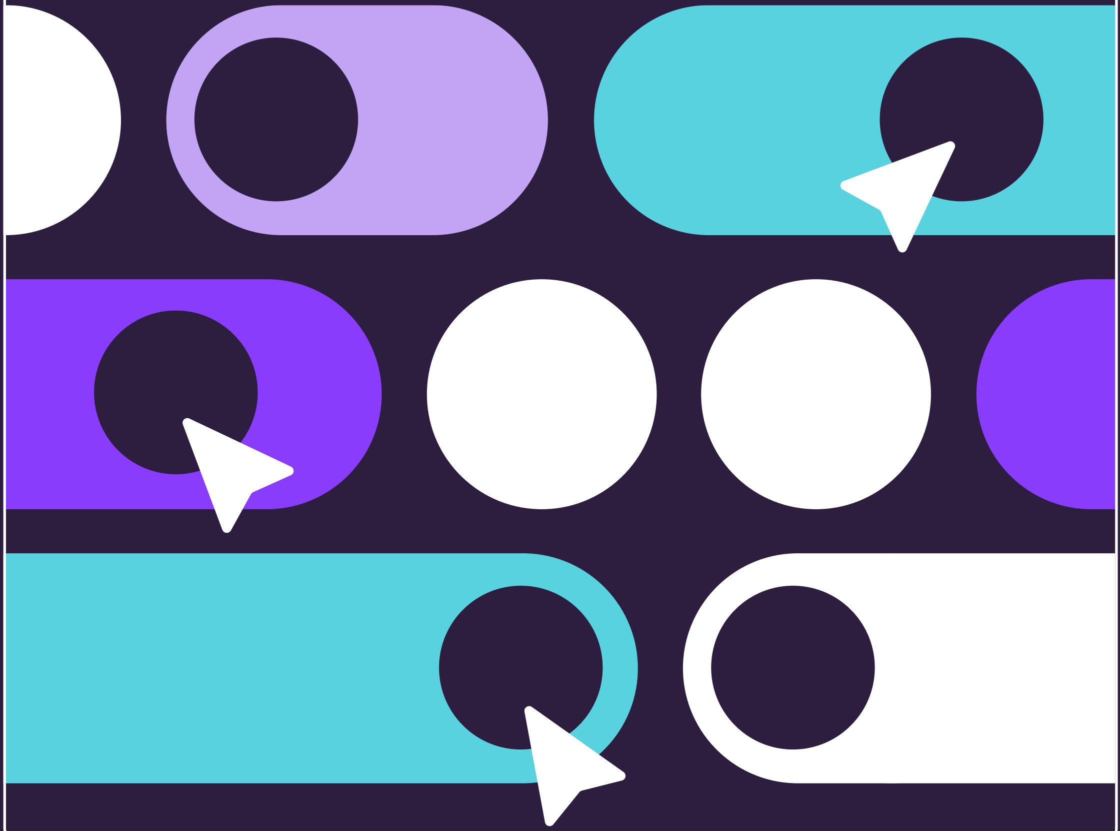
100M+

Developer workflows
automated with LinearB

LinearB

Learn more at linearb.io

The state of AI readiness



[INSIGHT 01] AI policy alignment is sharply polarized: 30.5% strongly agree they have a clear policy, while 21.1% strongly disagree.

[INSIGHT 02] With over 1 in 7 respondents (15.6%) strongly stating that their data isn't usable for AI, data quality was reported as the single most common structural blocker to AI adoption.

The State of AI readiness

	Strongly agree	Somewhat agree	Neither agree no disagree	Somewhat disagree	Strongly disagree
Our organization has a clearly defined and well-communicated policy for AI usage and experimentation.	30.5%	29.5%	13.7%	21.1%	5.3%
Our internal data sources are high-quality, well-documented, and easily accessible for AI workflows.	11.5%	24%	22.9%	26%	15.6%
Our version control practices are mature and make it easy to track changes and roll back when needed.	56.2%	25%	9.4%	7.3%	2.1%
Our team consistently works in small, manageable batches to deliver changes efficiently.	22.1%	45.3%	13.7%	13.7%	5.3%
Our developers are user-focused and prioritize delivering customer value in their work.	26.3%	46.3%	15.8%	10.5%	1.1%
Our internal tools and platforms are easy to use, reliable, and help reduce complexity in our workflows.	22.3%	34%	24.5%	13.8%	5.3%

AI readiness refers to an organization's ability to safely and effectively operationalize AI across the software delivery lifecycle. It depends not just on mature engineering habits, but on the foundational systems (data, platforms, tooling, and governance) that enable AI to become a force multiplier.

To assess this, our AI in Engineering Leadership Survey introduced an AI Readiness Matrix, modeled after DORA's seven categories of success criteria for AI adoption.

THIS FRAMEWORK EVALUATES WHETHER COMPANIES HAVE THE CORE CAPABILITIES NEEDED TO MAKE AI EFFECTIVE AT SCALE, COVERING SEVEN AREAS:

01	**Clarity of AI policies**
02	**Data quality**
03	**Version control maturity**
04	**Iterative delivery habits**
05	**User-focused development**
06	**Effectiveness of internal tooling**
07	**Platform reliability**

Because many organizations tend to overestimate their readiness, this matrix helps surface the structural gaps that can make or break AI adoption. To capture a true picture of confidence and alignment across teams, we used a Likert-style scale, asking respondents how strongly they agreed with each statement.

From a high level, our research revealed that engineering teams report strong maturity in their day-to-day practices: from disciplined version control (56.2%) to iterative workflows (67.4%) and a clear focus on user value (72.6%). However, the organizational systems meant to support AI (high-quality data, reliable internal tools, and clear strategic alignment) lag behind. When up to 65% of companies lack dependable data and nearly half doubt their tools or strategic direction, it's no surprise the environment fails to keep pace with the maturity of their teams.

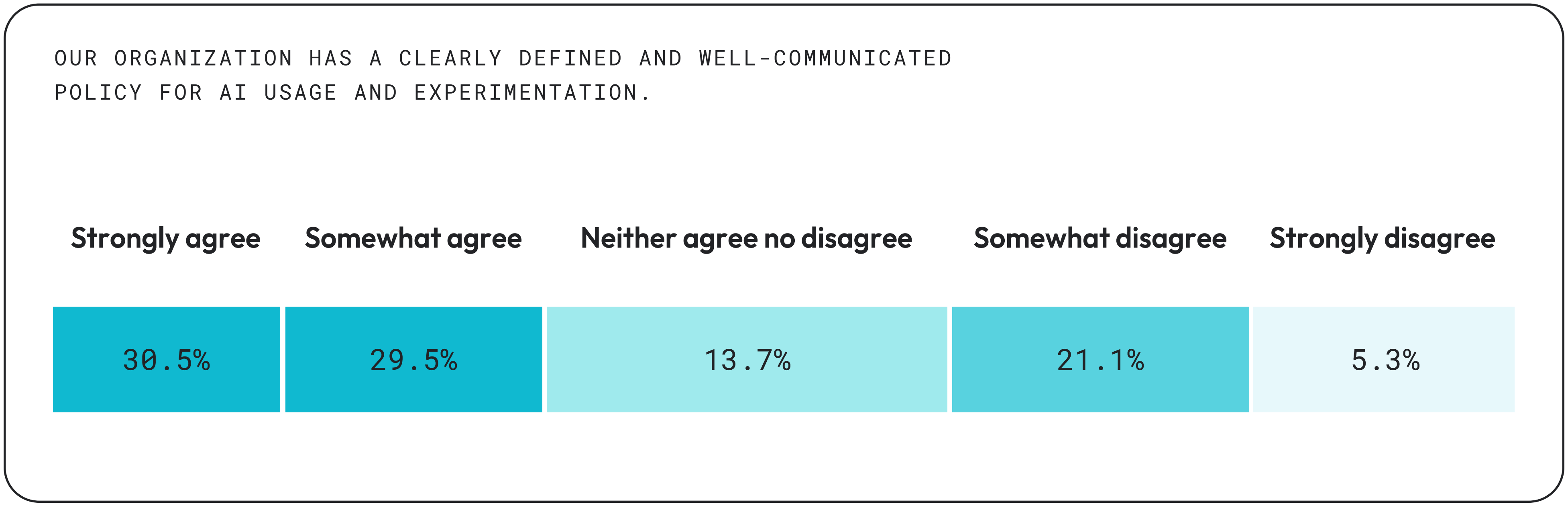
This gap suggests that while engineering talent and habits are not the limiting factors, foundational infrastructure (data quality, tooling, and organizational alignment) has become the new bottleneck to high-performing, AI-ready software delivery.

[INSIGHT 01]

AI policy alignment is sharply polarized — 30.5% strongly agree they have a clear policy, while 21.1% strongly disagree.

Responses to the first statement ('Our organization has a clearly defined and well-communicated policy for AI usage and experimentation') revealed one of the most polarized distributions in the entire survey. 30.5% of respondents strongly agree their organization has a clear AI policy, while 21.1% strongly disagree, a sharp split that shows teams clustering at opposite ends of the confidence spectrum, with very few in the middle.

This kind of bimodal response pattern suggests that AI policy is not evolving uniformly across engineering organizations. Some companies have already established explicit guidelines, communication channels, and governance structures. Others are still operating without a shared understanding of acceptable AI usage, whether due to slow-moving policy processes, lack of ownership, or simply being early in their AI journey. The risk, however, is straightforward: without a clear policy, AI adoption defaults to individual interpretation, leading to inconsistent practices, unmitigated security and compliance risks, and uneven performance outcomes across teams.



	Strongly agree	Somewhat agree	Neither agree no disagree	Somewhat disagree	Strongly disagree
Our organization has a clearly defined and well-communicated policy for AI usage and experimentation.	30.5%	29.5%	13.7%	21.1%	5.3%
Our internal data sources are high-quality, well-documented, and easily accessible for AI workflows.	11.5%	24%	22.9%	26%	15.6%
Our version control practices are mature and make it easy to track changes and roll back when needed.	56.2%	25%	9.4%	7.3%	2.1%
Our team consistently works in small, manageable batches to deliver changes efficiently.	22.1%	45.3%	13.7%	13.7%	5.3%
Our developers are user-focused and prioritize delivering customer value in their work.	26.3%	46.3%	15.8%	10.5%	1.1%
Our internal tools and platforms are easy to use, reliable, and help reduce complexity in our workflows.	22.3%	34%	24.5%	13.8%	5.3%

[INSIGHT 02]

With over 1 in 7 respondents (15.6%) strongly stating that their data isn't usable for AI, data quality was reported as the single most common structural blocker to AI adoption.

When respondents were asked whether their internal data sources are "high-quality, well-documented, and easily accessible for AI workflows," the results revealed that 15.6% of engineers "strongly disagree" with that statement, the highest "strongly negative" response across the entire readiness matrix. And when you add the "somewhat disagree" group (26%) and the "neither agree nor disagree" group (22.9%), the majority of respondents (64.5%) indicate that their data is not ready for AI at all.

This matters because AI readiness lives or dies on the quality of internal data. Even the most advanced models can only produce reliable outputs if the underlying data is accurate, discoverable, and complete. If data is scattered across systems, poorly documented, or hard to access, AI workflows immediately break down: models hallucinate, recommendations degrade, and engineering teams lose trust in AI-generated insights. The risk is not hypothetical. Without consistent, high-quality internal data, organizations simply cannot operationalize AI beyond isolated experiments.

Conclusion

What's next for AI?

The 2026 Software Engineering Benchmarks make one thing clear: AI is now embedded in everyday engineering work, but the surrounding practices, tooling, and organizational structures haven't fully caught up. Nearly 90% of developers use AI regularly, yet the technology's immaturity shows up across the delivery lifecycle: larger PRs, low Acceptance Rates, delayed and often shallow reviews, and unclear ownership for getting things done.

The qualitative feedback from engineering leaders explains why. AI-generated code often arrives without sufficient context, varies in quality, and introduces subtle risks that demand more verification. At the same time, teams are increasingly leaning on AI to summarize code, reduce review load, and accelerate delivery. This tension between speed and uncertainty is shaping how AI fits into modern workflows.

Looking ahead, our research shows that leaders expect AI to fundamentally reshape the nature of engineering work over the next 12 months. Many anticipate moving further up the abstraction ladder (toward architecture, orchestration, and system design) as AI handles more of the "grunt work." Others foresee dramatic productivity gains, deeper automation, and end-to-end agentic workflows. Some expect AI to help modernize legacy systems, improve quality, or streamline operational tasks. Yet a meaningful subset anticipates little change, signaling uneven readiness and maturity across organizations.

What engineering leaders are saying

We asked engineering leaders: ‘Looking ahead, how do you expect AI to influence your work in the next 12 months?’ Here’s what they shared:

“I expect to continue to have AI do the grunt work, and reserve my time for higher level processing around architecture.”

“All aspects of my role are changing to one of an “agent orchestrator”.”

“We’ll most likely be able to get way more done faster.”

“Expect to speed up productivity and delivery time by 20%.”

“Hoping that it can help us get out of legacy systems and away from places that suck the life out of our developers.”

“Exactly the same.”

“Don’t expect AI to influence my work.”

What becomes clear is that AI’s impact will not be uniform. It will amplify the strengths of organizations with solid processes, healthy data foundations, and effective internal tooling – and expose the weaknesses of those without them.

As engineering leaders enter this next phase, their challenge will be less about adopting AI and more about integrating it sustainably: developing guardrails, maturing review practices, measuring real impact, and building the cultural trust necessary for AI to deliver on its promise.

This year’s benchmarks aren’t a story of AI replacing how software is built – they’re a snapshot of teams reshaping their workflows around a technology that is powerful, widely adopted, and still evolving. The organizations that gain the most from AI will be the ones that invest in the structures needed to use it responsibly and effectively.

THE SHARE OF DEVELOPERS WHO USE AI REGULARLY IS NEARLY **90%**

Appendix

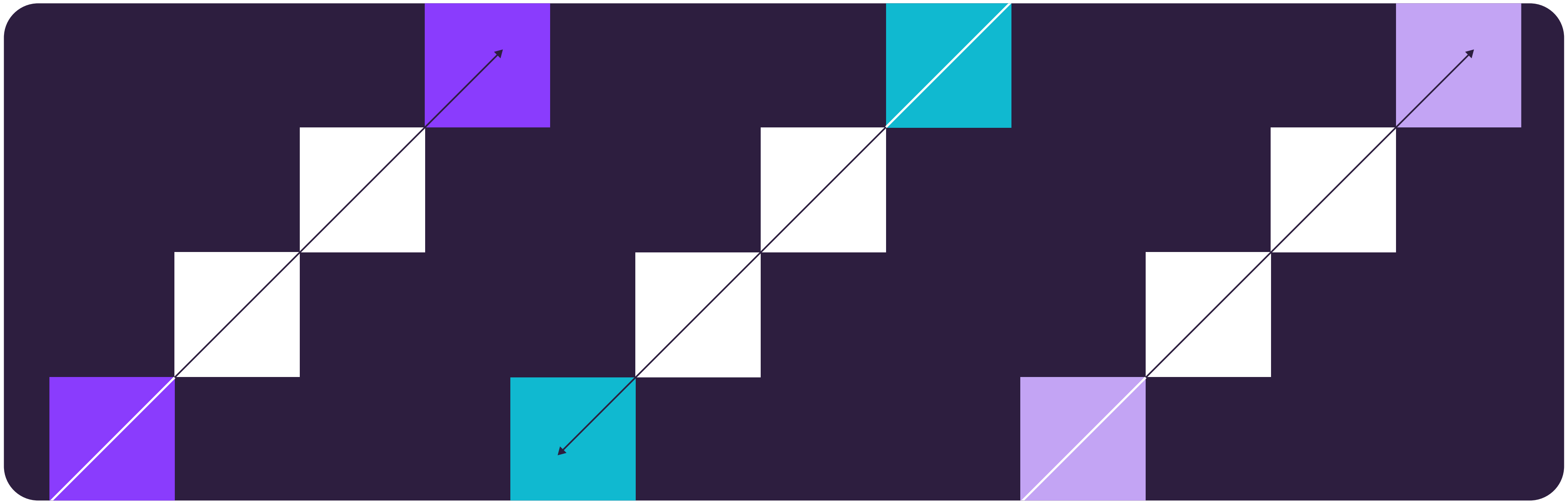
Where did MTTR go?

This year, we made the decision to remove MTTR (Mean Time to Restore) from our published benchmarks; not because these metrics lack value, but because the underlying data variance is too large to support responsible comparison organizations.

This variance is not a representation of a sudden shift in operational performance, but a fundamental aspect of how companies define and calculate MTTR. Some teams measure recovery time from the moment an issue is detected, others from when it's acknowledged, and still others from when customer impact is confirmed.

Some include partial degradations; others only include full outages. These disparate interpretations create data that looks comparable on the surface but is not methodologically aligned.

Rather than report metrics that could mislead our customers or create false comparisons, we chose to omit MTTR from the published benchmarks and recommend that organizations focus on their internal trends for this metric, rather than comparing to the industry. Our goal is to maintain benchmarks that are both trustworthy and actionable, and this year, the variability behind this metric made that impossible to guarantee.



Predictability benchmarks

Capacity Accuracy

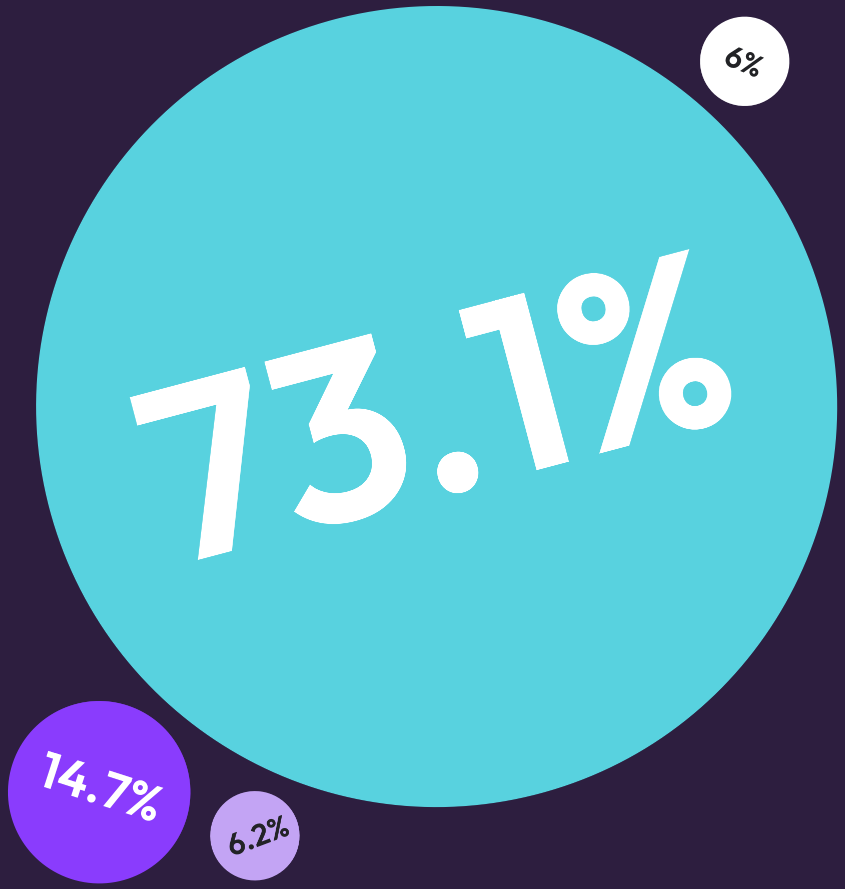
Measures how many issues (or story points) a team completed in an iteration (planned or unplanned) compared to the amount planned for that iteration.

Capacity Accuracy is a unique benchmark that is based on project data. It helps answer these questions:

- "Are teams taking on an amount of work that they can reasonably accomplish in an iteration?" Low Capacity Accuracy may point to over-commitment during planning.
- Are teams "playing it safe" and are under-committing to always hit the mark? This would lead to high Capacity Accuracy.
- Do we have a good grasp of the team's work capacity and can predictably deliver that capacity sprint over sprint? Unstable Capacity Accuracy that swings from low to high shows that the team hasn't found its pace yet and cannot predict the amount of work they can deliver.

ACTUAL DISTRIBUTION FOR CAPACITY ACCURACY

●	Ideal range	85% - 115%
●	Under commit	Above 130%
●	Potential under-commit	116% - 130%
●	Potential over-commit	70% - 84%



Planning Accuracy

The ratio of planned work vs. what is actually delivered during a sprint or iteration. High Planning Accuracy signals a high level of predictability and stable execution.

In a single “accuracy score” you’ll know if your teams are scoping iterations well, whether they’re completing their tasks, and how unplanned work is affecting execution.

The calculation is based on the following types of work:

- **Planned:** Story points or issues added before or within 24 hours of a sprint beginning.
- **Added:** Story points or issues added and completed after the sprint begins.
- **Completed:** Planned story points or issues completed in a sprint.
- **Carryover:** Planned but not completed story points or issues.

In some cases, teams may have a very high Planning Accuracy (> 95%), as well as a high Capacity Accuracy (> 130%). That means they aren’t taking on enough planned work, which should be a priority for improvement initiatives.

To begin accurately forecasting project delivery and determine if timelines can be moved up, we recommend tracking quality and predictability metrics like Planning Accuracy and Capacity Accuracy.

In our research, we found that a large majority of engineering teams (73.1%) under-commit their iteration plans, signaling a widespread tendency to play it safe during sprint planning. Only 14.7% of teams fall within the ideal Planning Accuracy range, where commitments align with delivery. Meanwhile, a combined 12.2% land in the potential over- or under-commit zones, indicating that relatively few teams are calibrating their capacity with the precision needed for predictable delivery.

Delivery metrics can be very useful early indicators for predictability. Tracking and improving metrics such as PR Size, Code Rework, and Cycle Time can help teams improve how they make and keep accurate promises to the business about delivery timelines.

PLANNING ACCURACY [AVERAGE]

ELITE	GOOD	FAIR	NEEEDS FOCUS
> 82%	82% - 64%	63% - 47%	< 47%

[AVERAGES]

Software engineering benchmarks

BENCHMARKS BY ORG | 8,109,244 PULL REQUESTS | 4,813 TEAMS | 163,820 ACTIVE CONTRIBUTORS

Category	Metric	Elite	Good	Fair	Needs focus
Delivery	Coding Time (hours)	< 19	19 - 36	36 - 70	> 70
	Pickup Time (hours)	< 8	8 - 14	15 - 22	> 22
	Approve Time (hours)	< 15	15 - 28	29 - 46	> 46
	Merge Time (hours)	< 7	7 - 13	14 - 24	> 24
	Review Time (hours)	< 11	11 - 20	21 - 38	> 38
	Deploy Time (hours)	< 21	21 - 82	83 - 181	> 181
	Cycle Time (hours)	< 45	45 - 83	84 - 150	> 150
	Merge Frequency (per dev/week)	> 2.0	2 - 1.2	1.2 - 0.66	< 0.66
	Deploy Frequency (per service)	> 1.2	1.2 - 0.5	0.5 - 0.2	< 0.2
	PR Size (code changes)	< 225	225 - 370	371 - 698	> 698
	PR Maturity (%)	> 89%	89 - 83%	82 - 77%	< 77%
Predictability	Change Failure Rate (%)	< 1%	1 - 4%	5 - 17%	> 17%
	Refactor Rate (%)	< 11%	11 - 16%	17 - 22%	> 22%
	Rework Rate (%)	< 3%	3 - 5%	6 - 8%	> 8%
	Capacity Accuracy (%)	85 - 115%	75 - 85% or 115 - 125%	70 - 75% or 125 - 130%	> 70% or > 130%
	Planning Accuracy (%)	> 82%	82% - 64%	63% - 47%	< 47%
Project Management	Issues Linked to Parents (%)	> 90%	90 - 67%	66 - 56%	< 56%
	Branches Linked to Issues (%)	> 77%	77 - 62%	61 - 41%	< 41%
	In Progress Issues with Estimation (%)	> 55%	55 - 26%	25 - 14%	< 14%
	In Progress Issues with Assignees (%)	> 96%	96 - 84%	83 - 76%	< 76%

[P50]

Software engineering benchmarks

BENCHMARKS BY ORG | 8,109,244 PULL REQUESTS | 4,813 TEAMS | 163,820 ACTIVE CONTRIBUTORS

Category	Metric	Elite	Good	Fair	Needs focus
Delivery	Coding Time (mins)	< 4 mins	4 - 7 mins	8 - 22 mins	> 22 mins
	Pickup Time (hours)	< 11 mins	11 - 26 mins	27 mins -1 hour	> 1 hour
	Approve Time (hours)	<1	1 - 2	3 - 6	> 6
	Merge Time (hours)	< 14 mins	14 - 30 mins	31 mins - 1 hour	> 1hr
	Review Time (hours)	< 21 mins	21 mins - 1 hour	1 - 2	> 2
	Deploy Time (hours)	< 1	1 - 24	25 - 133	> 133
	Cycle Time (hours)	< 2	2 - 15	16 - 30	> 30
	Merge Frequency (per dev/week)	> 2.0	2 - 1.2	1.2 - 0.66	< 0.66
	Deploy Frequency (per service)	> 1.2	1.2 - 0.5	0.5 - 0.2	< 0.2
	PR Size (code changes)	< 22	22 - 33	34 - 51	> 51
	PR Maturity (%)	> 89%	89 - 83%	82 - 77%	< 77%
Predictability	Change Failure Rate (%)	< 1%	1 - 4%	5 - 17%	> 17%
	Refactor Rate (%)	< 11%	11 - 16%	17 - 22%	> 22%
	Rework Rate (%)	< 3%	3 - 5%	6 - 8%	> 8%
	Capacity Accuracy (%)	85 - 115%	75 - 85% or 115 - 125%	70 - 75% or 125 - 130%	> 70% or > 130%
	Planning Accuracy (%)	> 82%	82% - 64%	63% - 47%	< 47%
Project Management	Issues Linked to Parents (%)	> 90%	90 - 67%	66 - 56%	< 56%
	Branches Linked to Issues (%)	> 77%	77 - 62%	61 - 41%	< 41%
	In Progress Issues with Estimation (%)	> 55%	55 - 26%	25 - 14%	< 14%
	In Progress Issues with Assignees (%)	> 96%	96 - 84%	83 - 76%	< 76%

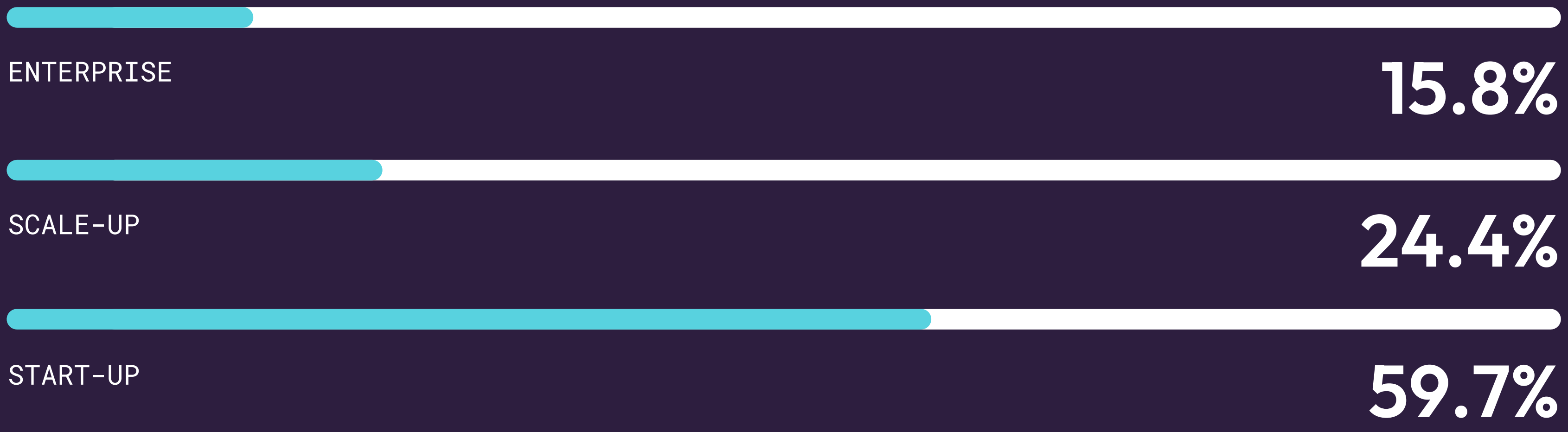
[P90]

Software engineering benchmarks

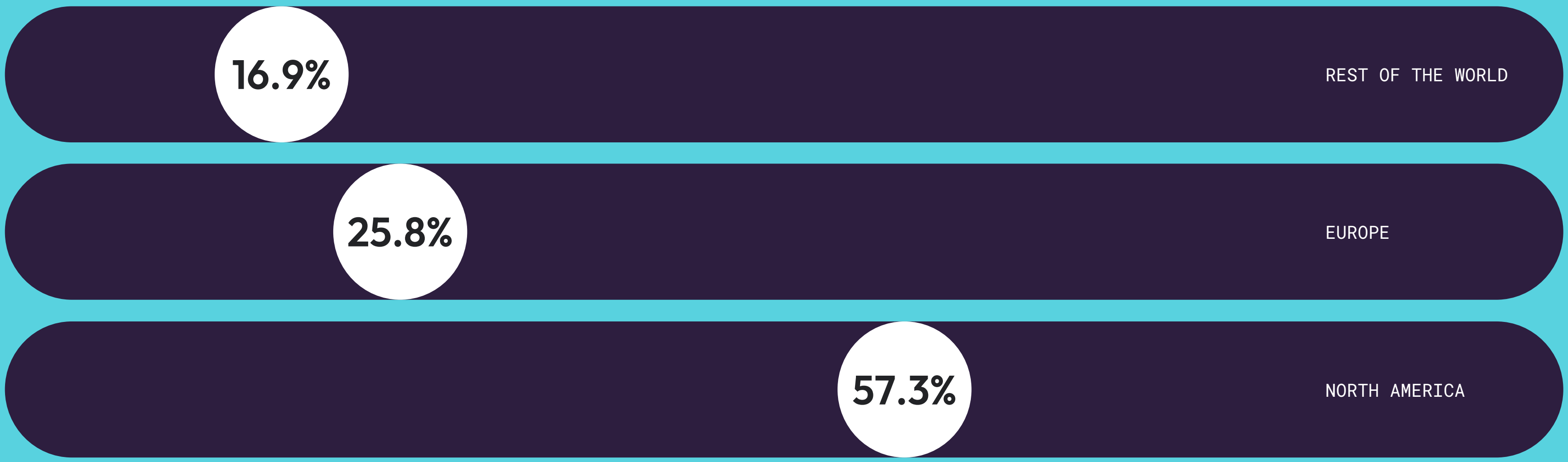
BENCHMARKS BY ORG | 8,109,244 PULL REQUESTS | 4,813 TEAMS | 163,820 ACTIVE CONTRIBUTORS

Category	Metric	Elite	Good	Fair	Needs focus
Delivery	Coding Time (hours)	< 26	26 - 73	74 - 151	> 151
	Pickup Time (hours)	< 16	16 - 25	26 - 63	> 63
	Approve Time (hours)	< 35	35 - 75	76 - 119	> 119
	Merge Time (hours)	< 16	16 - 23	24 - 66	> 66
	Review Time (hours)	< 22	22 - 53	54 - 97	> 97
	Deploy Time (hours)	< 43	43 - 195	196 - 455	> 455
	Cycle Time (hours)	< 114	114 - 192	192 - 388	> 388
	Merge Frequency (per dev/week)	> 2.0	2 - 1.2	1.2 - 0.66	< 0.66
	Deploy Frequency (per service)	> 1.2	1.2 - 0.5	0.5 - 0.2	< 0.2
	PR Size (code changes)	< 344	344 - 518	519 - 772	> 772
	PR Maturity (%)	> 89%	89 - 83%	82 - 77%	< 77%
Predictability	Change Failure Rate (%)	< 1%	1 - 4%	5 - 17%	> 17%
	Refactor Rate (%)	< 11%	11 - 16%	17 - 22%	> 22%
	Rework Rate (%)	< 3%	3 - 5%	6 - 8%	> 8%
	Capacity Accuracy (%)	85 - 115%	75 - 85% or 115 - 125%	70 - 75% or 125 - 130%	> 70% or > 130%
	Planning Accuracy (%)	> 82%	82% - 64%	63% - 47%	< 47%
Project Management	Issues Linked to Parents (%)	> 90%	90 - 67%	66 - 56%	< 56%
	Branches Linked to Issues (%)	> 77%	77 - 62%	61 - 41%	< 41%
	In Progress Issues with Estimation (%)	> 55%	55 - 26%	25 - 14%	< 14%
	In Progress Issues with Assignees (%)	> 96%	96 - 84%	83 - 76%	< 76%

ORG SIZE DISTRIBUTION



REGION DISTRIBUTION



LinearB is an AI Productivity platform that helps enterprises connect visibility, governance, and automation so teams can ship AI-driven code at scale. AI changed how we build software. Now it's time to change how we deliver it.

Book a demo