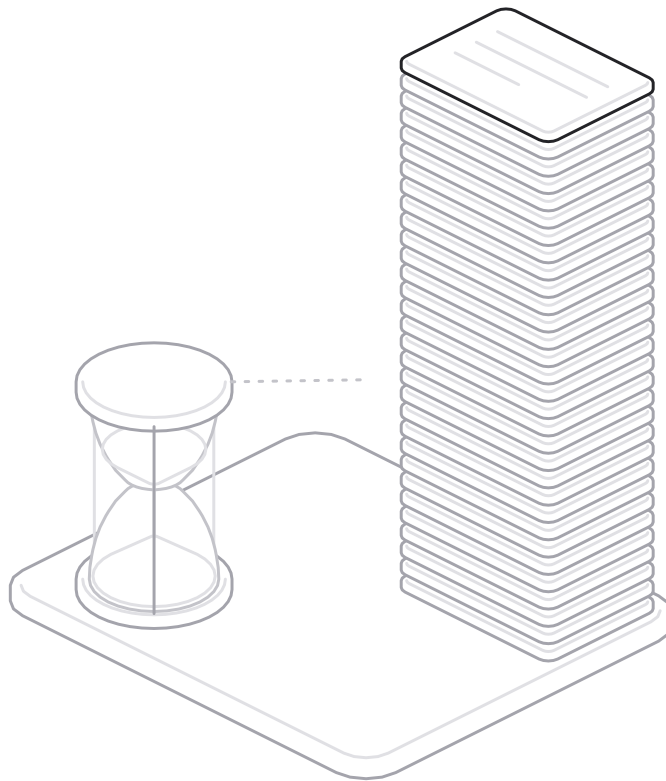


GUIDE

How to show that AI is working

What an engineering leader measures, and reports to the board, when the team builds with coding agents.

For engineering leaders who answer to a board, a CEO, or a CFO for their team's AI spend.



Contents

	In short	3
01	What do boards and CEOs ask when they ask if AI is working?	4
02	Our velocity went up 5x, but shipped value didn't change. Why?	4
03	What do DORA and SPACE say about measuring AI's impact?	6
04	Why are developers faster with AI while the team ships at the same pace?	7
05	Story points don't mean anything anymore with AI. How should we track productivity?	8
06	How do you calculate leverage without a dashboard?	9
07	Where do the other numbers come from?	11
08	What goes in a one-page board update on AI?	13
09	How do you roll out AI measurement in 30, 60, and 90 days?	14
10	What are the common mistakes when measuring AI productivity?	15
11	Where Hamster fits.	16
	Common questions	16
	Sources	17

Web version, with the templates to copy: tryhamster.com/guides/prove-ai-is-working

In short

Leverage is finished work per hour of attention, where finished work merged without coming back and attention is the time people spent thinking, aligning, briefing, steering agents, reviewing, and fixing.

-
- To show that AI is working, report leverage, the finished work a team ships per hour of attention, every month beside the customer outcomes that work moved, because leverage without an outcome is still activity.
-
- Token spend, lines of code, pull request counts, story points, and suggestion acceptance rates rise with agents whether or not the team ships more of what it agreed to build.
-
- Count review and rework as cost, because agents shorten the writing of code and the time moves to the people who review and fix it.
-
- Keep DORA's change fail rate and deployment rework rate beside leverage, so a team cannot raise the number by skipping review.
-
- Report one page to the board each month, in the same order, with the baseline, the coverage, and the customer or business outcome each shipped workstream moved.
-

Sooner or later a board member or a CEO asks whether the team's AI spend is working. Activity counts cannot settle it. Token spend, lines of code, pull request counts and story points all rise once agents arrive, whether or not the team ships more of what it agreed to build.

Leverage is the efficiency number to put in front of that board. Leverage is finished work per hour of attention. Finished work is work that merges without coming back, sized by the hours the team would have needed to build it without agents. Attention is the time people spend thinking, aligning, briefing, steering agents, reviewing their own work and other people's, and fixing what came back. At 1.0x, an hour of attention shipped an hour of work. Leverage rises only when the team does ship more, and it falls when speed piles up review and rework.

Leverage is half of the answer. The other half is the outcome the work was for: the customer or business result its Brief's Goals named, such as adoption, retention, revenue, support load, or

time to value. Leverage without an outcome is still activity, so this guide pairs the two throughout.

This guide is for the leader who has to answer the question every month. It expands the scorecard from Before the agents start into a method you can run with plain tools. You get a scorecard with leverage at the top, a worksheet to compute it, a one-page board update, and a 90-day rollout. The companion guide on raising team leverage covers what to change once you can see the number.

01

What do boards and CEOs ask when they ask if AI is working?

They ask whether the spend buys more of what the business needs, whether it costs reliability, and what the leader will change next. Few of them ask how many tokens the team used.

The question is fair, because most AI spending has not paid back yet. IBM surveyed 2,000 CEOs in 2025. They reported that 25% of their AI initiatives had delivered the expected return over the last few years, and 16% had scaled across the enterprise.¹ A board that has read numbers like these wants evidence from your team, in units it already uses.

Expect the question in some version of these four:

1. What are we getting for the AI spend?
2. Are we shipping more of what we agreed to build, or only more?
3. Did quality or reliability suffer?
4. What will you change next quarter, and what will it cost?

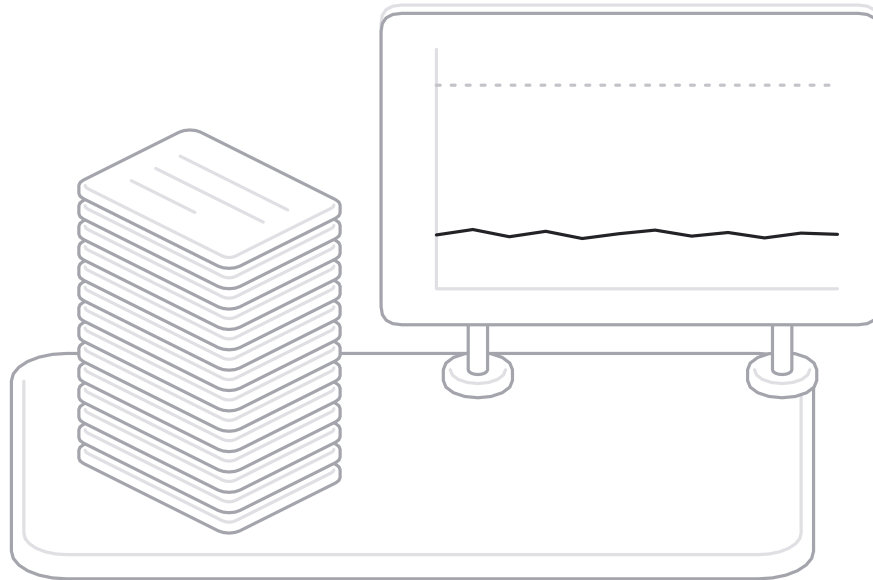
Each one has a measurable answer, and none of the answers is an activity count. Leverage and the outcomes the shipped work moved answer the first together. The share of finished work that traces to an agreed intent answers the second, and delivery stability answers the third. The fourth comes from the diagnostics that explain leverage, which show where the team's attention goes.

02

Our velocity went up 5x, but shipped value didn't change. Why?

Velocity, story points, pull request counts and lines of code all count activity, and agents produce activity cheaply. Work that merges and stays merged is still engineering output. Value arrives when that work reaches users and moves the customer or business outcome the team agreed to target.

Pull request counts show the gap most clearly. Faros looked at telemetry from more than 10,000 developers. On teams with high AI adoption, developers merged 98% more pull requests, while time spent in review rose 91%, average pull request size grew 154%, and bugs per developer rose 9%. Faros found no significant link between AI adoption and delivery outcomes at the company level.²



The most quoted AI number in public is the share of code a model wrote. On Alphabet's third-quarter 2024 earnings call, Sundar Pichai said "more than a quarter of all new code at Google is generated by AI, then reviewed and accepted by engineers."³ A share-of-code figure says how much code a model wrote. It says nothing about whether anyone needed that code, or what reviewing it cost.

Suggestion acceptance rate has the same weakness. In GitHub's study of its own code completion tool, the acceptance rate of shown suggestions was the best predictor of how productive developers felt.⁴ It predicts perception, which is a different thing from output.

Perception is unreliable on its own. In METR's randomized trial, experienced open-source developers took 19% longer to finish tasks with early-2025 AI tools. Afterward, they believed the tools had made them 20% faster.⁵ METR's 2026 update estimated an 18% speedup for the developers who returned, with a confidence interval running from a 38% speedup to a 9% slowdown, wide enough to include no effect.⁶

Developers took 19% longer with AI, and believed afterward they had been 20% faster.

METR, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity," July 2025

Every one of these numbers also bends once it becomes a target. Charles Goodhart observed in 1975 that "any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes."⁷ Marilyn Strathern's later phrasing is the one most people remember: "When a measure becomes a target, it ceases to be a good measure."⁸ A team asked for more pull requests will open more pull requests.

03

What do DORA and SPACE say about measuring AI's impact?

Both say to measure the delivery system with several numbers read together, and both warn against a single activity count. Leverage fits inside them, with DORA's stability metrics as its guardrail.

DORA measures software delivery with five metrics.⁹ Three describe throughput: change lead time, deployment frequency, and failed deployment recovery time. Two describe instability: change fail rate and deployment rework rate, which DORA defines as "the ratio of deployments that are unplanned but happen as a result of an incident in production."

DORA's surveys show why the stability side matters. Its 2024 report estimated that every 25% increase in AI adoption came with a 1.5% reduction in delivery throughput and a 7.2% reduction in delivery stability.¹⁰ By 2025 the throughput relationship had turned positive, while higher adoption still went with more delivery instability. The same report found that 90% of respondents use AI at work and 30% trust AI-generated code a little or not at all.¹¹

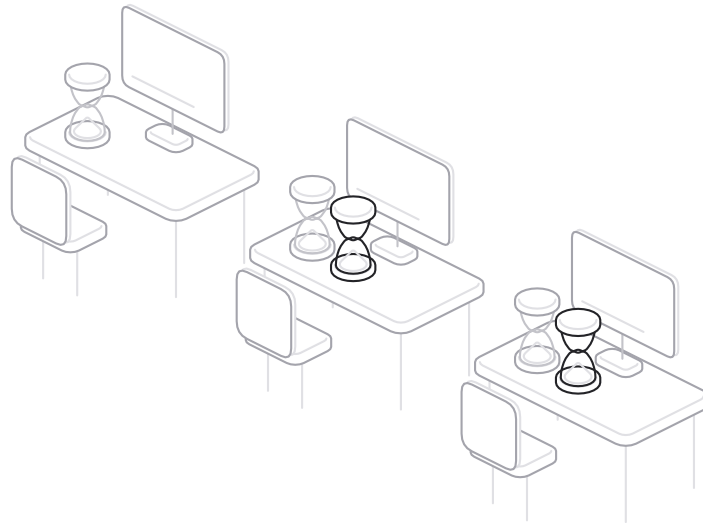
The SPACE framework names five dimensions of developer productivity: satisfaction and well-being, performance, activity, communication and collaboration, and efficiency and flow. Its authors wrote that developer productivity "cannot be measured by a single metric or dimension."¹²

Leverage respects both warnings. It puts output (performance) over attention (efficiency and flow), counts review of other people's work (collaboration), and ignores raw activity. It does not replace DORA's stability metrics, which belong beside it on every report.

04

Why are developers faster with AI while the team ships at the same pace?

The time moved. Agents shorten the writing of code, and the people around that code absorb the extra volume in review and rework, which most dashboards never count.



Controlled studies do find faster tasks. Google ran a randomized trial with 96 of its own engineers. Those with AI features finished an enterprise coding task about 21% faster, with a wide confidence interval.¹³ Field experiments at Microsoft, Accenture, and a Fortune 100 company found a 26.08% increase in completed tasks among developers given GitHub Copilot.¹⁴

The cost lands after the task. GitClear found that code churn, the share of new lines revised within two weeks of being written, rose from 3.1% in 2020 to 5.7% in 2024.¹⁵ Two in three developers say their biggest frustration with AI tools is output that is “almost right, but not quite,” and 45% say debugging AI-generated code takes more time.¹⁶ CodeRabbit’s review of open-source pull requests found about 1.7 times as many issues in AI-authored pull requests as in human-written ones.¹⁷

Rework was expensive before agents. Boehm and Basili’s widely cited estimate put avoidable rework at 40 to 50% of a software project’s effort.¹⁸ Agents make rework faster to produce and easier to miss, because the fix arrives as another pull request that looks like output.

So a measure of AI’s impact has to count review and rework as cost. Leverage does, because both are attention. Work that bounces back costs twice: the finished work arrives later, and the fix adds hours to the denominator.

Story points don't mean anything anymore with AI. How should we track productivity?

Track leverage, the finished work the team ships per hour of attention, and read it monthly beside the customer outcomes the work targeted and a few diagnostics that explain why it moved. Retire story points and velocity as evidence of AI's impact, because agents inflate both without changing what ships.

Story points estimate effort, and agents change the effort behind a point from one week to the next, so velocity drifts with the tools. Leverage keeps the size of the work fixed at what it would have cost the team without agents, and puts the hours people actually spent beside it.

TEMPLATE **Leader's scorecard**

Read monthly, over a rolling 30-day window. Report the team, never a ranking of people.

Headline

- Leverage: finished work divided by attention.
 - Finished work: work that merged without coming back, sized in the team-hours it would have taken without agents.
 - Attention: hours of thinking, aligning, briefing, steering agents, reviewing your own and others' work, and fixing. Meetings do not count.
 - 1.0x means an hour of attention shipped an hour of work.

Outcomes

- For each agreed workstream that shipped: the customer or business outcome its Goals named, the metric, the baseline, and this period's result against target.
- Leverage without an outcome is still activity.

Why it moved

1. Ship-ready rate: share of finished work that merged without going back for rework.
2. Attention by phase: hours in briefing, steering, review, rework, and reviewing others.
3. Agreed workstreams in flight: workstreams running now from an intent the affected people agreed to by name.
4. Rework by cause: hours spent fixing work that came back, split into intent wrong or missing, defect, and scope change. Report rework avoided against the baseline.
5. Alignment time: days from the first draft of an intent to agreement.
6. Intent coverage: share of merged work that links to an agreed intent.

Guardrails

7. Delivery stability: DORA change fail rate and deployment rework rate.
8. Spend: AI tool and model spend per hour of finished work.

Not evidence on their own: tokens used, lines of code, share of code written by AI, suggestion acceptance rate, pull request count, story points, self-reported time saved.

Leverage and the outcome results answer the board's first question together. Leverage says how much finished work each hour of attention bought, and the outcomes say whether that work changed anything for customers or the business. The diagnostics explain it, and each one points at a different place in the work. Ship-ready rate and rework by cause show whether agents build what was meant. Attention by phase shows where people's hours go, including the review a manager does for everyone else. Agreed workstreams in flight shows how much work runs at once from shared intent. Agents add capacity there, but parallel agents raise leverage only when their work merges without pulling people back in.

The guardrails keep the headline honest. A team can raise leverage for a month by skipping review, and the change fail rate will show it. Spend belongs beside output as a cost, never as a measure of progress.

Try it with your AI

Your coding agent can run this guide with you. Paste one line into Claude Code, Cursor, or Codex, in the repo your team works in.

Read <https://tryhamster.com/guides/prove-ai-is-working/skill.md> and run it with me.

1. It reads 90 days of merged pull requests, reverts, and reviews from your repo, and your Goals from Hamster.
2. It computes a first leverage figure with its coverage, and fills the scorecard, with every assumption marked.
3. It drafts the one-page board update for you to edit, and never ranks people.

06

How do you calculate leverage without a dashboard?

List the work that merged in the last 30 days and drop anything that came back. Size each item by what it would have cost without agents, add up the hours of attention the team spent, and divide the first total by the second.

TEMPLATE Leverage worksheet

Window: the last 30 days. Team: one team, not a ranking of people.

1. List finished work: every workstream or pull request that merged in the window.
2. Apply the 30-day tail: void any item reverted within 30 days of merge, and charge follow-up fixes to the item they fixed.
3. Size each item in team-hours the team would have needed without agents. Use the estimate made before work started. Never use pull request or line counts.
4. Collect attention per person, by phase: briefing, steering, review, rework, reviewing others. Leave meetings out.
5. $\text{Leverage} = \text{total size in team-hours} / \text{total attention hours}$.
6. Record coverage: the share of each person's working hours you captured as attention.

Item	Merged	Reverted or fixed within 30 days	Size (team- hours)	Briefing	Steering	Review	Rework	Reviewing others

Sizing is the weak point, so make it boring. Estimate each piece of work before agents start, in the hours the team would have needed by hand, and use the same method every month. A team that estimates generously will inflate its own number. Size a few pieces of work the team built without agents and compare, so the scale stays honest.

Attention can be captured in one of two ways. The first infers it from timestamps the team already produces: messages to agent sessions, pull request reviews and comments, commits, and edits to intent documents. Group each person's events into sessions, join events less than 20 minutes apart, add 5 minutes to each session, and count a lone event as 5 minutes. The second is a weekly log of five numbers per person, one per phase. A log is an estimate, so keep the method fixed and read the trend, not the level.

Charge rework to the work that caused it. A pull request reverted within 30 days of merge stops counting as finished work, and the hours spent on it stay in the denominator. A follow-up fix charges its hours to the original work. Speed that breaks things later then shows up on its own ledger, without anyone labeling it bad.

Rework avoided falls out of the same records. Take the rework hours per hour of finished work in your baseline, subtract this month’s rate, and multiply by this month’s finished work. The result is hours the team did not spend fixing, and it is the most concrete number you can give a CFO.

07

Where do the other numbers come from?

From three places: your git host, a record of agreed intent, and a label on every pull request that came back. Most teams have the first, few have the second, and the third takes a week to start.

TEMPLATE **Rework labels and PR template line**

Labels for pull requests that came back after review, or were reverted:

rework:intent The work matched the request, but the request was wrong or missing a decision.

rework:defect The work did not do what the agreed intent said.

rework:scope Someone changed what was wanted after agreement.

reverted Merged, then reverted within 30 days.

Line for the pull request template:

Intent: <link to the agreed Brief or intent document, and the version that was agreed>

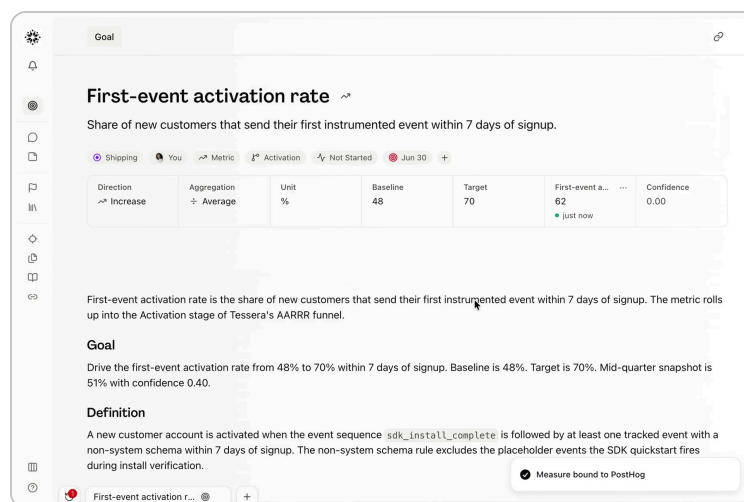
The rework labels matter more than they look. Rework caused by intent is the cheapest to avoid, because the fix happens before any agent starts: agree on what to build, by name, and record the decisions behind it. Before the agents start describes that step. Defects and scope changes have other causes, and mixing the three hides which one is growing.

In Hamster, intent and its agreement already live in one record. Each Brief carries the Ready and Not yet votes of the people whose work it changes, the Plan that agents build from, and the pull requests linked to it. The customer outcomes sit on the same chain, because each Brief belongs to an Initiative that links to the Goals it serves. Leverage can be tracked as a Goal whose Metric is tagged leading or lagging, with a baseline, a target, and a Result logged each period.

TEMPLATE **Where each number comes from**

Measure	Without Hamster	In Hamster today
Leverage	The leverage worksheet, monthly	The worksheet, with Brief and pull request records as inputs

Measure	Without Hamster	In Hamster today
Ship-ready rate	Pull requests with "changes requested", commits after review, reverts	Pull requests linked to each Brief, with their reviews
Attention by phase	Event timestamps grouped into sessions, or a weekly log	The worksheet
Agreed workstreams in flight	Intent documents with an agreement date and open pull requests	Briefs in Shipping with Ready votes, filtered by owner
Rework by cause	Rework labels on pull requests	Rework labels on pull requests
Alignment time	Date of first draft to date of agreement	Ready and Not yet votes, timestamped on each Brief's activity timeline
Intent coverage	The Intent line in the pull request template	Pull requests linked to Briefs
Delivery stability	Incident tracker and deploy tool	Incident tracker and deploy tool
Customer outcomes	Product analytics, revenue and support reports, matched to each workstream by hand	Goals with a Metric and a Result each period, linked through Initiatives to the Briefs that serve them; an actual can come from PostHog, GA4, Amplitude, or Mixpanel
Spend	Vendor invoices and usage exports	Vendor invoices and usage exports
The monthly read	A calendar reminder	A Goal whose Metric is leverage, with a Result each period, and a scheduled Routine that drafts the update



What goes in a one-page board update on AI?

One headline number with its trend, the customer outcomes the shipped work moved, the diagnostics that explain the number, the risks, and one change for next month. Keep the same order every month, so the board learns to read it.

TEMPLATE One-page board update

AI and engineering leverage, <month>

Headline: leverage was <x.x>x over the last 30 days, against <baseline>x before <start date>.
<One sentence on why it moved.>

Measure	This month	Last month	Baseline
Leverage			
Ship-ready rate			
Agreed workstreams in flight			
Rework hours avoided			
Change fail rate			
Deployment rework rate			
AI spend per hour of finished work			

What shipped and what it moved: <two or three agreed workstreams, each with the customer or business outcome its Goals named, the metric, and this period's result against target>

What moved the number: <the diagnostic that changed most, with the evidence>

Risks: <stability, review load, people>

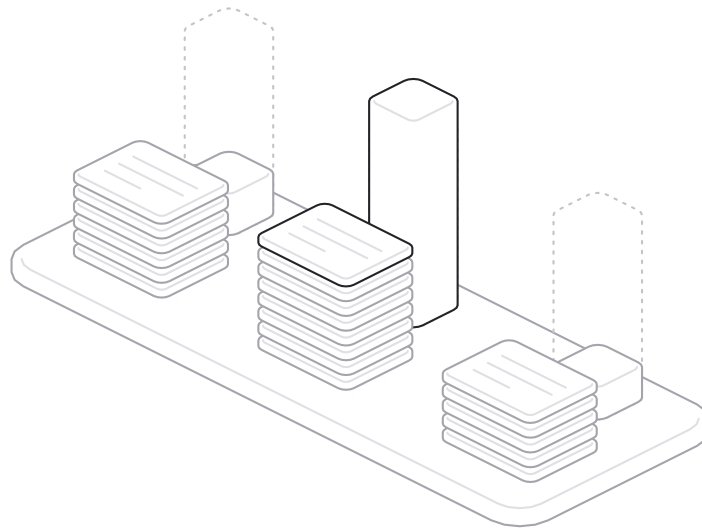
Next month: <one change, what it costs, and the measure it should move>

How this was measured: <share of attention captured, and how work was sized>

The last line earns the board's trust. A leverage figure built on half the team's hours is still useful, as long as the board knows it covers half. State the coverage and the sizing method every time, and change neither without saying so.

Name real work in "What shipped and what it moved." If a workstream shipped and its outcome has not moved yet, say so and give the date you expect to read it. A board remembers the workstream that moved a company goal far longer than a ratio. The link between the two shows that the AI spend paid for something the business wanted. In Hamster, a scheduled Routine can

draft this page on the same day each month from the Briefs and Plans that shipped, for you to check and send.



09

How do you roll out AI measurement in 30, 60, and 90 days?

Spend the first month on a baseline, the second on a first honest reading, and the third on the first report. Change nothing because of a single month.

TEMPLATE 30/60/90-day rollout

Days 1 to 30: baseline

- Write down the scorecard definitions, and keep them fixed for the quarter.
- Pull 90 days of merged pull requests. Mark reverts and follow-up fixes within 30 days.
- Label a sample of 30 pull requests that came back, by rework cause.
- Choose how to capture attention: event timestamps or a weekly log.
- Compute a first leverage figure, with its coverage.

Days 31 to 60: first reading

- Add the Intent line to the pull request template, and start the rework labels.
- Run two or three workstreams from intent the affected people agreed to by name.
- Read the scorecard at the end of the month.
- Draft the board update, and show it to one peer before anyone else.

Days 61 to 90: first report

- Compare the month against the baseline.
- Pick the diagnostic that drags most, and make one change to raise it.
- Send the board update.
- Decide whether to add more teams.

The baseline is the step teams skip, and without it every later number is an anecdote. Ninety days of merged pull requests, with reverts and follow-up fixes marked, is enough to compute a first leverage figure and a first ship-ready rate. It also shows how much rework the team carried before anything changed.

10

What are the common mistakes when measuring AI productivity?

The worst mistake is turning the scorecard into a target for individuals. The rest come from reading too little data, or from changing how it is counted.

- Do not rank people. Reviewing other people's work counts as attention, so it lowers the reviewer's own figure on purpose, which makes a manager's review load visible. A ranking would punish the people who review, so report the team and show each person only their own figure.
- Do not set targets on the diagnostics. A target on ship-ready rate invites smaller, safer work, and a target on alignment time invites agreement nobody means. Set the target on leverage, and use the diagnostics to find what to change.
- Keep the sizing method fixed for the quarter. A new method breaks the trend, so change it at a quarter boundary and restate the baseline.
- Wait before reading the latest month. Recent work keeps collecting follow-up fixes for 30 days, so each month settles over the next one.
- Leave out work that came back. A reverted pull request is not finished work, however fast it merged.
- State the coverage. If the method captured 20 of 40 working hours, the board should know.
- Describe what moved together, not what caused what. Most studies in this guide report associations, and your own before-and-after comparison is one too.
- Do not report leverage without outcomes. A rising number with no customer or business result behind it is still activity.

- Keep instability on the page. Leverage without the change fail rate invites the question you least want from a board.

11

Where Hamster fits.

A team that runs many agents at once is running a software factory, and Hamster makes that factory build what the team agreed before it builds anything. The team shapes intent together in a Brief, votes Ready or Not yet on it, and hands the agreed version to agents through a Plan. Agents in Claude Code, Cursor, and Codex read the Brief and the team's [Context Graph](#) and write their decisions back while they work, and pull requests link back to the Brief they came from.

That record makes leverage measurable, because every piece of finished work traces to an intent people agreed to. Every review and fix attaches to the work it served, and every decision is kept with who made it. The same record keeps the outcome beside the number. Each Brief belongs to an Initiative, each Initiative links to the Goals it serves, and each Goal's Results record the target and the actual every period.

You can run everything in this guide with a spreadsheet and your git host. If you want the record to stay live across every person and every agent session, Hamster was built for that.

Common questions

How do you measure whether AI is working in an engineering team?

Measure leverage, the finished work the team ships per hour of attention, over a rolling 30 days, and pair it with the customer outcomes the work targeted, such as adoption, retention, revenue, or support load. Read both beside ship-ready rate, attention by phase, rework by cause, and DORA's stability metrics.

Story points don't mean anything anymore with AI. How should we track productivity?

Track leverage instead. It sizes work by what it would have cost the team without agents and divides that by the hours people actually spent, so it does not drift with the tools.

Our velocity went up 5x, but shipped value didn't change. Why?

Velocity, story points, and pull request counts measure activity, and agents produce activity cheaply. Value arrives only when merged work that stays merged reaches users and moves the customer or business outcome the team set out to change.

How often should an engineering leader report leverage to the board?

Read it monthly and report it on one page, in the same order each time. Put the baseline and the coverage beside it, and name the work that shipped against a company goal.

Should we report the share of code written by AI?

No. A share-of-code figure says how much code a model wrote, not whether anyone needed it or what reviewing it cost.

What should a board update on AI include?

One headline leverage figure with its trend and baseline, the customer outcome each shipped workstream moved, the diagnostics that explain the number, the risks, and one change for next month. State how much of the team's time the figure covers.

Can you calculate leverage without a dashboard?

Yes. List the work that merged in 30 days, drop anything reverted, size each item in team-hours without agents, add up the hours of attention, and divide.

Sources

1. IBM Institute for Business Value, 2025 CEO Study, May 2025. Survey of 2,000 CEOs. <https://newsroom.ibm.com/2025-05-06-ibm-study-ceos-double-down-on-ai-while-navigating-enterprise-hurdles>
2. Faros AI, The AI Productivity Paradox, 2025. Telemetry from more than 10,000 developers on 1,255 teams; correlational. <https://www.faros.ai/blog/ai-software-engineering>
3. Sundar Pichai, remarks on Alphabet's third-quarter 2024 earnings call, October 2024. <https://blog.google/inside-google/message-ceo/alphabet-earnings-q3-2024/>
4. Albert Ziegler and colleagues, "Productivity Assessment of Neural Code Completion," MAPS 2022. <https://arxiv.org/abs/2205.06537>
5. METR, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity," July 2025. <https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/>
6. METR, "We are Changing our Developer Productivity Experiment Design," February 2026. <https://metr.org/blog/2026-02-24-uplift-update/>

7. Charles Goodhart, "Problems of Monetary Management. The U.K. Experience," Papers in Monetary Economics, Reserve Bank of Australia, 1975.
8. Marilyn Strathern, "'Improving ratings'. Audit in the British University system," European Review, 1997, p. 308.
9. DORA, "DORA's software delivery performance metrics." <https://dora.dev/guides/dora-metrics/>
10. DORA (Google Cloud), Accelerate State of DevOps Report, 2024. Reported as an association, not a cause. <https://cloud.google.com/blog/products/devops-sre/announcing-the-2024-dora-report>
11. DORA (Google Cloud), State of AI-assisted Software Development, 2025. <https://dora.dev/research/2025/dora-report/>
12. Nicole Forsgren, Margaret-Anne Storey, and colleagues, "The SPACE of Developer Productivity," ACM Queue, 2021. <https://queue.acm.org/detail.cfm?id=3454124>
13. Elise Paradis and colleagues, "How much does AI impact development speed? An enterprise-based randomized controlled trial," Google, 2024. <https://arxiv.org/abs/2410.12944>
14. Zheyuan Kevin Cui, Mert Demirer, and colleagues, "The Effects of Generative AI on High-Skilled Work. Evidence from Three Field Experiments with Software Developers." https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4945566
15. GitClear, AI Copilot Code Quality, 2025. Share of changed lines, 2020 to 2024. https://www.gitclear.com/ai_assistant_code_quality_2025_research
16. Stack Overflow Developer Survey 2025, AI section. <https://survey.stackoverflow.co/2025/ai>
17. CodeRabbit, State of AI vs Human Code Generation Report, December 2025. <https://www.coderabbit.ai/blog/state-of-ai-vs-human-code-generation-report>
18. Barry Boehm and Victor R. Basili, "Software Defect Reduction Top 10 List," IEEE Computer, 2001. <https://www.cs.umd.edu/~basili/publications/journals/J81.pdf>



Run this with your team.

Hamster is free for 10 Briefs a month, with unlimited viewers.

tryhamster.com/get-started

This guide on the web: tryhamster.com/guides/prove-ai-is-working