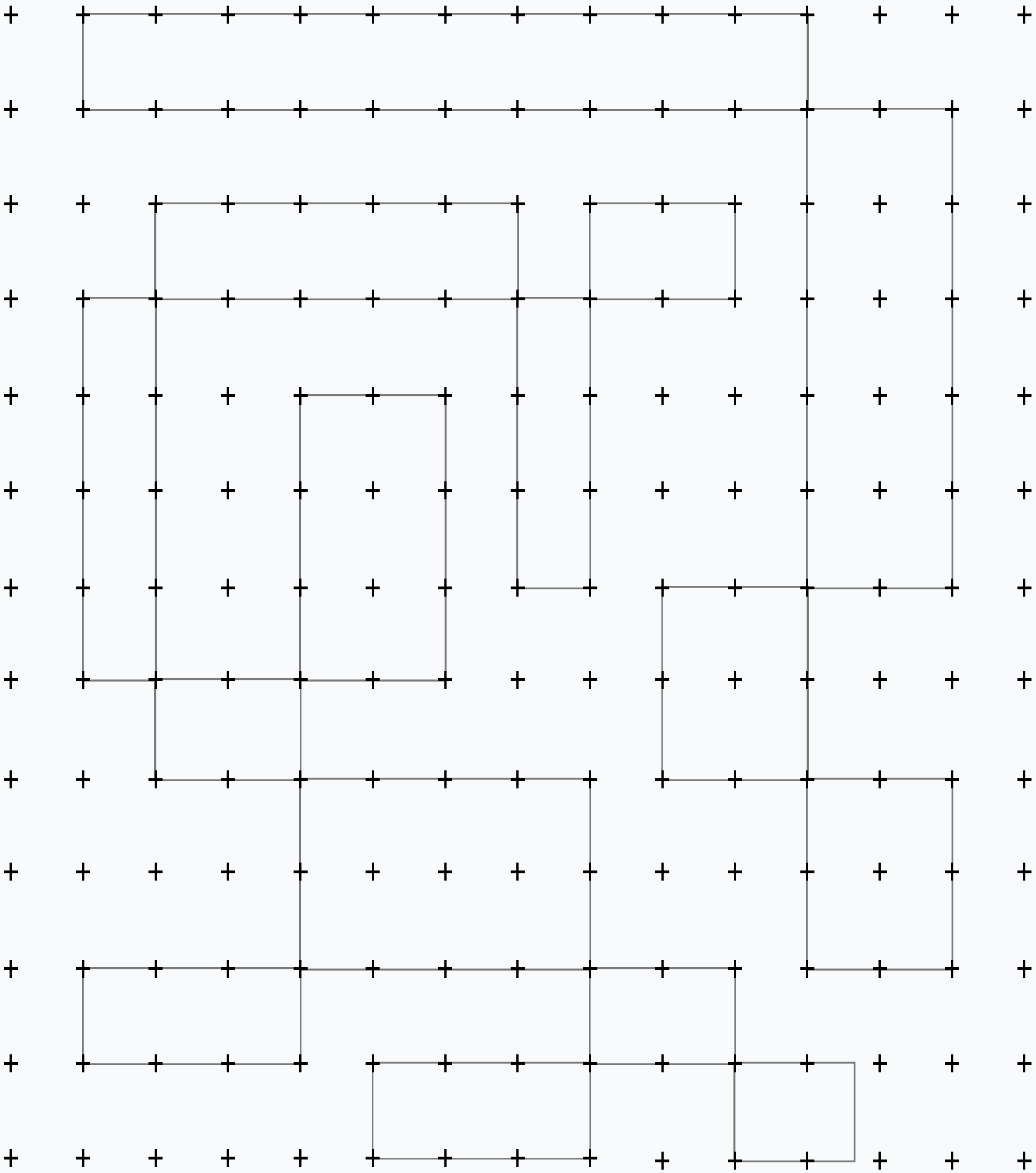
EXPERT GUIDE

Developer Productivity Metrics at Top Tech Companies



At DX, we regularly get asked to share what other companies are doing to improve developer productivity. A common form of this request is a leader asking which developer productivity metrics to adopt within their organization.

We often work with Developer Productivity teams, which require metrics in order to fully understand the developer experience at their company and track changes to productivity. These teams think a lot about metrics. We believe that other leaders can learn from what these teams have already learned and which metrics they're finding genuinely valuable.

To provide insight, we interviewed leaders from Developer Productivity functions at 17 top tech companies. The metrics they shared are listed in the following table, and defined in the final section of this report.

Developer Productivity Metrics at 15+ Top Companies

Company	Name of function	What they measure
Amplitude	Platform Engineering	Ease of Delivery, Engineer Engagement, Deploys per Service, Change Failure Rate
ATLASSIAN	Developer Experience	DSAT (Developer Satisfaction), Time from Commit to Deploy, Pull Request Cycle Time, Self-Serve Documentation, Self-Serve Dependency Maintenance
chime	Engineering Operations	Developer CSAT, Perceived Rate of Delivery, Ease of Delivery, Sentiment and Quantitative Metrics for 16+ DevEx factors
DOORDASH	Developer Platform	Developer Satisfaction, Adoption Rate, Stability of Services/Apps
Etsy	Enablement	Experiment Velocity, Availability, Performance, Developer Sentiment
GoodRx	Engineering Solutions	Engagement, Weekly Time Loss, Deep Work, Ease of Delivery, Speed, Stability, % Adoption of Standards
Google	Engineering Productivity	Speed, Ease, and Quality *Specific metrics for each of these dimensions vary
INTERCOM	Developer Experience	Ease of Delivery, Perceived Productivity, Engineer Engagement
Lattice	Developer Experience	Perceived Ease of Delivery, CI Pipeline Failure Rate, CI Pipeline P50 Build Time, Number of Deployments, Change Failure Rate
LinkedIn	Productivity & Happiness	Developer Build Time, Code Reviewer Response Time, Post-Commit CI Speed, CI Determinism, Deployment Success Rate, Developer NSAT (Net User Satisfaction)
Microsoft	Engineering Thrive	SPACE metrics *Specific metrics for each of these dimensions vary
Notion	Developer Infrastructure	Developer survey with a focus on perceptions of friction
PELOTON	Tech Enablement	Developer Satisfaction Score, Time to 1st and 10th PR, Lead Time, Deployment Frequency, % of PRs under 250 lines, Line Coverage, Change Failure Rate, Time to Restore Services
POSTMAN	Platform Engineering	Ease of Delivery, Perceived Productivity, Weekly Time Loss, Engineer Engagement, Cycle Times, Satisfaction
Spotify	Developer Experience	Self-Reported Productivity, % Adoption to Golden Standards, Deployment Frequency, Weekly Deployments per Weekly Active Developer
stripe	Developer Tools	Avg. number of days with Sufficient Focus Time, Branch Creation to Master Merge Time, Sentiment, Weekly # of PRs per Developer
Uber	Developer Platform	Weekly # of Diffs/PRs per Engineer, Weekly # of Code Reviews per Engineer, Weekly # of Design Docs Generated per Engineer, Avg. Weekly Focus Time per Engineer

What Top Companies Measure

The table provides an overview of developer productivity metrics used across different companies. In this section, we'll zoom in to take a closer look at the metrics used by several organizations that are different in size.

Google (10k+ employees)

Google has a Developer Intelligence team dedicated to measuring developer productivity and providing insights to leaders across the company. For instance, they help internal tooling teams understand how developers are using their tools, whether they're satisfied with them, and how fast, reliable, or intuitive the tools are. They also partner with executives to understand the productivity of their organizations.

Whether measuring a tool, process, or team, Google's Developer Intelligence team subscribes to the belief that no single metric captures productivity. Instead, they look at productivity through three dimensions: speed, ease, and quality. These dimensions exist in tension with one another, helping surface potential tradeoffs.

To illustrate, consider the example of measuring the code review process. The team would capture metrics for each of the three dimensions:

- Speed: How long does it take for code reviews to be completed?
- Ease: How easy or difficult is it for developers to move through the code review process?
- Quality: What is the quality of feedback received from a code review?

Again, this is just one example that uses code review as a way to illustrate. The specific metrics that Google uses varies depending on the subject of measurement. But their three core dimensions remain constant.

To calculate metrics, Google relies on a mix of both qualitative and quantitative methods to get the fullest picture possible. Ciera Jaspan, tech lead manager within the Developer Intelligence team, explains: "We will measure using logs for speed. We'll also measure people's beliefs of how fast they think they're going. We will also follow this up with diary studies and interviews to make sure that this all lines up and matches up together. We're talking about mixed methods."

Many of the metrics that Google uses are captured through behavioral methods. Collin Green, UX research lead and manager within the Developer Intelligence team, explains: “Technical debt is a thing that we've run into that is just hard to find good objective metrics that tell you how much and where and whether it's a problem. Surveys can help you measure things that you don't know how to measure objectively. It can also help you measure things that are in principle not measurable objectively.”

LinkedIn (10k+ employees)

LinkedIn, like Google, has a centralized Developer Insights team that’s responsible for measuring developer productivity and satisfaction, and delivering insights to the rest of the organization. This team sits within the broader Developer Productivity and Happiness organization, which focuses on reducing friction from key developer activities and improving the internal tools they use.

LinkedIn uses three channels to capture developer productivity metrics:

- Quarterly survey: The Developer Insights team uses a quarterly survey to assess the developer experience across a range of tools, processes, and activities. It includes approximately 30 questions, which developers answer in about 10 minutes. The survey is delivered through a proprietary platform developed and maintained by the Developer Insights team, allowing for advanced customization and personalization of survey questions based on data collected from their real-time feedback and metrics systems.
- Real-time feedback system: To capture feedback in between quarterly surveys, LinkedIn has developed a real-time feedback system. This system tracks events and actions that developers perform within development tools, and sends targeted surveys based on specific triggers. The system uses smart throttling mechanisms to avoid overwhelming developers with feedback requests.
- System-based metrics: LinkedIn also calculates metrics using data from their systems, providing high precision measurements for things such as build times and deployment frequency. The Developer Insights team maintains a global system for ingesting and analyzing this data, which they call the Developer Insights Hub (or iHub). This system allows teams across LinkedIn to create custom dashboards and metrics tailored to their needs.

Here are some examples of the specific metrics that they focus on:

- Developer Net User Satisfaction (NSAT) measures, on a quarterly basis, how happy developers are overall with LinkedIn's development systems.
- Developer Build Time (P50 and P90) measures the time, in seconds, developers spend waiting for their builds to finish locally during development.
- Code Reviewer Response Time (P50 and P90) measures how long it takes, in business hours, for code reviewers to respond to each update of the code review from the author.
- Post-Commit CI Speed (P50 and P90) measures how long it takes, in minutes, for each commit to get through the continuous integration (CI) pipeline.
- CI Determinism is the opposite of test flakiness: the likelihood that a test suite's result will be valid and not a flake.
- Deployment Success Rate measures how often deployments to production succeed.

LinkedIn's Developer Insights team looks at both qualitative and quantitative metrics in each of these areas. For example, for build times, they'll compare the objective measure of how long builds take with how satisfied developers are with their builds. Grant Jenks, Senior Tech Lead for their developer insights platform, explains: "Even if the quantitative metrics say that everyone's builds are fantastic, if developers are saying 'I hate my builds,' you should probably listen to that."

While the quantitative metrics listed above are typically calculated using medians (i.e., 50th percentiles), one challenge they've found with medians is that metrics can be less moveable when outliers are improved. For example, in one instance, excessively long front-end builds were reduced from 25 seconds to 3 seconds. But because the median time was higher than that, the impact wasn't being reflected in their metrics.

To solve these issues, they often use winsorized means which are calculated by replacing high and low end values with numbers closer to the middle. Jenks explains: "Figure out your 99th percentile. And instead of throwing away all the data points that are above the 99th percentile, clip them. So if your 99th percentile is like 100 seconds, and you have a data point that's 110 seconds, you cross out 110, and you write 100. And now you calculate your mean."

Although not listed in the table, a special metric LinkedIn provides to teams is a composite score called the Developer Experience Index which is an aggregate score based on a number of different individual metrics such as the ones listed earlier. Jenks generally cautions against the use of composite scores due to the complexity of developing and calibrating them. He explains: “we don't track it over time. We reserve the right to change the aggregation and the weightings behind it at any time. We tell people don't ever put this in an OKR.”

Peloton (1-10k employees)

Peloton’s measurement approach began by capturing qualitative insights through developer experience surveys. Later, they started pairing this data with quantitative metrics to derive a more full picture.

Peloton measures productivity by focusing on four key areas: engagement, velocity, quality, and stability. Here are some of the metrics they use to capture each:

- Engagement: Developer Satisfaction Score
- Velocity: Time to 1st and 10th PR for all new hires, Lead Time, Deployment Frequency
- Quality: % of PRs under 250 lines, Line Coverage, Change Failure Rate
- Stability: Time to Restore Services

Thansha Sadacharam, head of tech learning and insights at Peloton explains the roots of their survey program: “I very strongly believe, and I think a lot of our engineers also really appreciate this, that engineers aren't robots, they're humans. And just looking at numbers or looking at certain key metrics don't drive the whole story. So for us, having a really comprehensive survey that helped us understand that entire developer experience was really important.”

Their survey is conducted biannually, with each survey being sent to a random sample of roughly half of their developers. With this approach, individual developers only need to participate in one survey per year, minimizing the overall time spent on filling out surveys while still providing a statistically significant representative set of data results.

Scaleups (500-1k employees)

There are several scaleups on the list: Amplitude, GoodRx, Intercom, Lattice, Notion, and Postman. Here are some commonalities in what these companies measure:

1. Ease of Delivery: Most of these companies measure ease of delivery, a qualitative measure of how easy or difficult developers feel it is to do their work.

Multiple DevProd leaders shared how they use this metric as a "north star" for their work, since their teams' goal is to make developers' lives easier. This metric is also useful as a way to show the impact of these teams thanks to it being fairly moveable (i.e., directly impacted by the work of these teams). From a theory standpoint, this metric also captures key aspects of the developer experience such as cognitive load and feedback loops.

2. Engagement: Most of these companies also track engagement, a measure of how excited and stimulated developers feel with their work. While engagement is commonly measured in HR engagement surveys, DevProd teams also cited focusing on Engagement for several reasons. First, developer engagement and productivity are closely linked. In other words, "happy developers are productive developers" and so developer engagement can be viewed as an indicator of productivity. In addition, leaders mentioned that a key benefit of measuring engagement is to counterbalance other metrics which emphasize speed.

3. Time Loss. GoodRx and Postman pay attention to the average amount of lost time, measured by the percentage of developers' time that is lost due to obstacles in their work environment. This metric is similar to ease of delivery in that it provides DevProd teams a moveable metric that can be directly impacted by their work. In contrast to ease of delivery, one of the benefits of measuring time loss is that it can be well-translated into dollars, and therefore easily understood by other business leaders. For example, if an organization with $10,000,000 in fully loaded engineering payroll costs is able to reduce time loss from 20% to 10% through an initiative, that would translate into $1,000,000 of savings.

4. Change Failure Rate. Change Failure Rate – one of the four key metrics from the DORA research program – is a top-level metric tracked by several companies including Amplitude and Lattice. The DORA team defines change failure rate as the "percentage of changes to production or releases to users result in degraded service (for example, lead to service impairment or service outage) and subsequently require remediation (for example, require a hotfix, rollback, fix forward, patch)."

Lattice measures Change Failure Rate as the number of PagerDuty incidents divided by the number of deployments. Amplitude measures it as the P0s over production deploys (the P0 count goes through PagerDuty, and the deploy count is from Spinnaker).

Other unique metrics

As we've seen, there's quite a bit of overlap in what different developer productivity teams are measuring. But there are also a few unique metrics worth highlighting:

- **DoorDash, GoodRx, and Spotify all track versions of Adoption Rate.** Spotify's version is a measure of how many developers have adopted their Golden Standards. In general, this is a measure of how many developers actively use a product or service.
- **Uber's Design Docs Generated per Engineer.** Design docs are written by engineers for non-trivial projects before they start meaningful work—the idea is to get feedback early and ultimately decrease the time taken to complete a project. Their metric tracks how frequently developers are following this practice.
- **Etsy's Experiment Velocity.** At Etsy, each team designs and runs its own experiments to assess how users will respond to new features. This practice is a core aspect of their engineering culture, facilitating a culture of learning and helping teams stay focused on the customer. Etsy has developed an in-house experimentation platform to track the progress of these experiments. Metrics include how many experiments are started each week, how many are stopped, and how many had a positive hit rate. For context, Former CTO, Mike Fisher, has said the ultimate goal would be to measure learning velocity.
- **Chime and LinkedIn's Developer CSAT/NSAT.** Chime measures a Developer Customer Satisfaction (CSAT) score for every tool and service developers use. This metric is captured through their quarterly developer surveys. LinkedIn's Developer NSAT (Net User Satisfaction) measures how satisfied developers are overall with their development systems. These metrics are different in two ways: first, Chime's CSAT focuses on specific tools, whereas LinkedIn's NSAT measures developers' satisfaction with all tools overall. Also, Chime's metric is calculated as a percentage of positive responses, whereas LinkedIn's is the percentage of satisfied responses subtracted by the percentage of dissatisfied responses.

How to choose your own metrics

We generally recommend following Google's Goals, Signals, Metrics (GSM) framework to help guide metric selection. Too often, teams jump to specific metrics before thinking through what they truly want to understand or track. The GSM framework can help teams decide what their goal is, and then work backwards to select metrics that serve their goal.

Ciera Jaspan, from Google's Developer Intelligence team, has explained how the GSM process is used at Google: "We always encourage people to follow the goal, signals, metrics approach. We ask them to first write down your goals. What is your goal for speed? What is your goal for ease? What's your goal for quality? Write those down first and then ask your question of, what are the signals that would let you know that you've achieved your goal? Regardless of whether they're measurable. Signals are not metrics. What would be true of the world if you've achieved your goal? At that point, try to figure out what are the right metrics."

Here are some examples that may help:

If you're a Developer Productivity team

Most DevProd teams have a charter that sounds something like "make software engineering easier at our company." For example, the charter for Google's Developer Insight team is to make it fast and easy for developers to deliver great products. Slack's is to "make the development experience seamless for all engineers," and Stripe's is to "make software engineering easier."

We can work backwards from these goals to define our metrics. If we want to make it easier for developers to deliver high quality software, how do we know whether we've done that? We might look for signals such as:

- How easy it is for developers to deliver software
- How quickly developers are delivering software
- The quality of software that's being delivered

For each of these categories, we can define metrics to help us know how we're doing. For example:

- Speed = Perceived Delivery Speed, Perceived Productivity
- Ease = Ease of Delivery, Deployment Lead Time, Build Failure Rate
- Quality = Incident frequency, Perceived Software Quality

These metrics should sound similar to many of the metrics discussed earlier in this article. Top-level metrics like these can help your DevProd team convey the value and impact of your efforts and keep everyone aligned both within and outside of your team.

In addition to top-level metrics, DevProd teams also need operational metrics to tie to specific projects or OKRs. These could be metrics like developer satisfaction with specific tools, adoption rate of a particular service, or granular measurements of developers' workflows. There are no good one-size-fits-all solutions here, but the important thing is for teams to choose metrics they can control, rather than trying to target high-level key metrics that can be affected by a number of confounding factors.

If you're an executive

If you're a CTO, VPE, or Director of Engineering, it's likely that your scope is broader than the definition of developer productivity that's been discussed thus far in this article. In fact, when I speak with engineering leaders who are figuring out metrics, they've often been asked for metrics by their CEO or leadership team. My advice here is to reframe the problem. What your leadership team wants is less about figuring out the perfect productivity metrics and much more about being made to feel confident that you're being a good steward of their investment in engineering.

To demonstrate good stewardship, consider selecting metrics that fall within three buckets:

1. Business impact. You should report on current or planned projects alongside data that addresses questions like: Why are these the right things to build now? How does this project make the business money or otherwise support its goals? Is this project on track or delayed?

This type of reporting is often seen as the responsibility of the product team, but only engineering can represent the full set of projects being worked on. (For example, a database migration project is unlikely to be on product's roadmap).

2. System performance. Engineering organizations produce software, so stakeholders will want to know about the health and performance of these systems. Are they fast and reliable? Secure and well-maintained? Are users satisfied with them? Useful metrics to report here include things like uptime, number of incidents, and product NPS scores. If you have a dedicated Infra or UXR team, they likely are capturing metrics that fall under this bucket.

3. Engineering effectiveness. Stakeholders want to know how effective the engineering organization is and how it can be improved. This article has been primarily focused on this bucket, so you can take what we've learned from how DevProd functions measure and apply it here.

Business impact What are we working on right now, and why?	**System Performance** How performant and reliable are our systems?	**Developer Effectiveness** How effective is the engineering organization and how can it be improved?
Current and planned projects, including: • Business rationale and success metrics • Project status (e.g. delivered, on track, at-risk)	• Application latency and uptime • Number of incidents • Server costs • User NPS	Measures of speed, ease, and quality, for example: • Perceived Rate of Delivery • Weekly Time Loss • Ease of Delivery • Developer Satisfaction • Incident Frequency • Perceived Software Quality • Change Failure Rate

Metrics definitions

Once you've determined your goals for measuring, use this list of metrics as a starting point for selecting and defining the specific metrics you'll use. This is a list of all the metrics that were surfaced by the companies mentioned in this article.

Adoption Rate
Adoption rate is a measure of how many developers actively use a product or service. Ideally, this is a percentage of the number of developers using the product out of the total number of developers that the product is intended to serve. However, some teams use simpler metrics: for example, calculating the total number of users that have ever used a product. They may also calculate the total number of users that have used the product within a given timeframe, such as within the past month or quarter.

There are also examples of companies measuring the adoption of a process, such as Uber measuring "Design Docs Generated per Engineer." Design docs are written by engineers for non-trivial projects before they start meaningful work: the idea is to get feedback early and ultimately decrease the time taken to complete a project. Uber's metric tracks how frequently developers are following this practice.

Availability
Availability measures the percentage of time that your infrastructure is operational and accessible within a given time period. It's often used as a metric to report on when discussing system performance.

Change Failure Rate
One of the four key DORA metrics, change failure rate is used as a measure of stability. The DORA team defines change failure rate as "percentage of changes to production or releases to users result in degraded service (for example, lead to service impairment or service outage) and subsequently require remediation (for example, require a hotfix, rollback, fix forward, patch)." The difficult part of this metric is defining "failure."

For some real-world examples, Lattice measures Change Failure Rate as the number of PagerDuty incidents divided by the number of deployments. Amplitude measures it as the P0s over production deploys (the P0 count goes through PagerDuty, and the deploy count is from Spinnaker). Another way to calculate Change Failure Rate is to measure the percentage of deployments that were hotfixes or rollbacks.

CI Determinism (CID)
CI Determinism is the opposite of test flakiness: it measures the likelihood of a test suite's result being valid, and not a flake. The benefit of using this metric over Test Flakiness is that CI Determinism is a number that's good when it goes up.

LinkedIn is the only company listed that includes CI Determinism as a top-level metric. They use a system that runs CI tests at specific times every week to track whether these tests give consistent results or whether they change from one run to another. Each test gets a score based on how often it gets the same result. If a test is run 10 times and it passes 7 out of those 10 times, its Determinism Score would be 70%. A higher score is better because it means the test is more reliable. When they aggregate the metric, they average all the Determinism Scores to get an overall Determinism Score. This way, codebases that run less frequently but are still flaky have their flakiness equally represented in the metric as codebases that are run frequently.

Code Reviewer Response Time
This measures how long it takes for code reviewers to respond to each update from a developer during a code review. Both LinkedIn and Google subscribe to the belief that one of the most important qualities of a code review is that reviewers respond quickly. This metric measures how quickly reviewers respond to each update that a developer posts.

LinkedIn calculates Code Reviewer Response Time as the time, in business hours, that it takes between each request and response. A request is when a reviewer gets a notification that an author has taken some action, and now the author is blocked while waiting for a response. "Response" is defined as the first time after a request that a reviewer or code owner responds to the PR and sends that response to the author. LinkedIn specifically looks at the P50 and the P90 values.

As a similar metric, Atlassian measures Pull Request Cycle Time (also known as PR Creation to Merge Time). Theirs is a measure of the average time it takes for a pull request to go from 'open' to 'merged' over the last ten pull requests. The difference between Atlassian's and LinkedIn's metric is that Atlassian's looks at the full process, whereas LinkedIn is focused on driving a specific behavior (faster response times).

Deep Work
Time for Deep Work, or "Focus Time," is a measure of the amount of uninterrupted time developers have at work. Most teams use surveys to capture variations of this metric:

- Developer satisfaction with the amount of time they have for deep work can be measured in a survey, with the question: "How satisfied are you with the amount of uninterrupted time you have for deep work?"
- Meeting Heavy Days or the inverse, Number of Days with Sufficient Focus Time, can be tracked by asking developers: "In a typical week, how many days do you have with more than one scheduled meeting (not including standups)?" Response items should provide a scale of options, such as 0 days, 1 day, 2 days, 3 days, 4 or more days.
- Interruption Frequency can be measured by asking developers: "In a typical week, how often are you interrupted from your primary task to work on something else that was unplanned or suddenly requested?" Response items should provide a scale of options, such as Less than once per week, At least once per week, At least once every two days, At least once per day, At least once every couple of hours.

Deployment Frequency
Deployment Frequency is another one of the four key DORA metrics: it measures how often an organization successfully releases to production. Teams may choose to look at the frequency of successful deployments over any given time period (hourly, daily, weekly, monthly, yearly). The challenge with this metric is defining what constitutes a successful deployment to production. DORA's research looks at successful deployments to any amount of traffic, but teams may define a successful deployment differently (for example, deploying to 50% or more traffic).

Measuring deployment frequency in a generic way can be tricky for some organizations. As an alternative, LinkedIn shared that they've begun to work on a metric called "Deployment Freshness," which measures how old code is in production. Improving Deployment Freshness should bring the same value to the business as improving Deployment Frequency.

Developer Build Time
Developer Build Time measures how much time developers spend waiting for their build tool to complete. This is a common metric that Developer Productivity teams focus on because it often represents a big opportunity to improve developer productivity. In many companies, developers spend a significant amount of their time waiting for builds to complete, and even small improvements to make builds faster are beneficial.

LinkedIn defines this as the wall-clock time from when the build tool starts a "build" to when it completes. The duration is measured and reported in seconds. Critically, LinkedIn only counts this only for builds invoked by human beings, that we reasonably assume they are waiting on (this is notable because other teams have run into issues by including build times from robotic builds in their metric). LinkedIn excludes all builds run on the CI infrastructure in this metric.

Developer Customer Satisfaction (CSAT) and Net User Satisfaction (NSAT)
Developer Satisfaction metrics can capture how satisfied developers are overall with their development systems, or how satisfied they are with specific tools. Satisfaction is typically captured quarterly in a developer experience survey. Teams can ask a question about developers' overall satisfaction, and also ask about their experience using specific tools (CSAT).

Some Developer Productivity teams also measure Engagement, which can be captured in a developer survey with the question "How energized are you by your work?". This is a measure of how excited and stimulated developers feel with their work. It's commonly measured in HR engagement surveys, however DevProd teams also focus on engagement because it is an indicator of productivity. Additionally, it can be used to counterbalance other measures that emphasize speed. Delivering software faster is good, but not at the expense of developer happiness.

Ease of Delivery
Many of the companies included in the report measure Ease of Delivery, which is a qualitative measure of how easy or difficult it is for developers to do their work. This is frequently used as a "north star metric" for Developer Productivity teams since their mission is to make it easier for developers to do their jobs.

Ease of Delivery can be captured in a quarterly survey using the question, "How easy or difficult is it for you to do work as a developer or technical contributor at [Company]?"

Experiment Velocity (or Learning Velocity)
Experiment Velocity is a unique metric from Etsy. At Etsy, experiments are a core aspect of their engineering culture as a way to bring teams closer to the customer and learn quickly. Each team at Etsy designs and runs its own experiments to assess how users will respond to new features.

The Experiment Velocity metric was developed using an in-house experimentation platform that tracks the progress of these experiments. Etsy pays attention to how many experiments are started each week, how many are stopped, and how many had a positive hit rate.

Lead Time for Changes
Another one of the four key DORA metrics, Lead Time for Changes measures the amount of time between a code change and the release of this change to end users. This is a measure of speed.

As described by the DORA program, "The Lead Time for Changes metric requires two important pieces of data: when the commit happened, and when the deployment happened. This means that for every deployment, you need to maintain a list of all the changes included in the deployment. This is easily done by using triggers with a SHA mapping back to the commits. With the list of changes in the deploy table, you can join back to the changes table to get the timestamps, and then calculate the median lead time."

GitLab measures Lead Time for Changes by calculating the median time it takes for a merge request to get merged into production (from master).

Perceived Rate of Delivery
Perceived Rate of Delivery is a measure of how fast or slow a developer feels their team delivers software. This is a measure of speed.

Teams typically use Perceived Rate of Delivery to understand whether development teams feel as though they're delivering quickly or not. When teams only rely on quantitative metrics, they often wonder whether what they're seeing is good or bad. Take deployment frequency: this metric alone doesn't tell us how difficult it is for a team to deploy code, or whether a team feels they're shipping software quickly. Perceived Rate of Delivery provides that data.

Perceived Productivity is another similar metric used for the same reasons: it provides the developers' perspective on how often they feel productive within a given week.

Time to Restore Service
Time to Restore Service is another one of the four key DORA metrics. It measures how long it takes an organization to recover from a failure in production. It's intended to be used as a measure for stability.

This metric is typically paired with Change Failure Rate which measures the percentage of changes that require a rollback or hotfix. Time to Restore services will track the time from when the change (that required a rollback) was released, to when it was resolved.

Atlassian measures MTTR as the average time it takes to fully resolve a failure, as measured from when an incident occurs to when the component becomes fully functional again, measured over the last 10 incidents. GitLab measures Time to Restore Service as the median time (number of seconds) an incident was open for on a production environment, in the given time period they're viewing.

Time to 1st and 10th PR
Time to 1st and 10th PR is a measure Peloton uses to understand the ramp-up time for new developers. These metrics aren't used to evaluate individual developers, but rather to measure the impact of improvements to the onboarding process, which is something their Tech Enablement & Developer Experience team has focused on.

Weekly Time Loss
Weekly Time Loss calculates the percentage of time that developers lose due to obstacles or inefficiencies in their work environment (for example, slow tools or processes, unplanned work, unclear tasks). Similar to Ease of Delivery, this is frequently used as a "north star metric" by Developer Productivity teams to track the impact of their work. For example, if they introduce a change that brings Weekly Time Loss from 23% down to 20%, that can translate to a significant impact on the business for a mid-sized organization or larger.