

WHITEPAPER

Simple Metrics to Measure Engineering Efficiency

How to avoid painful delays and costly budget overruns **with the right frameworks**

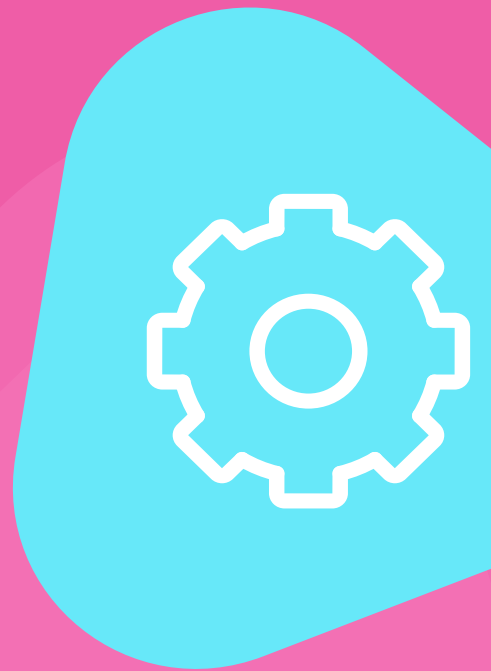
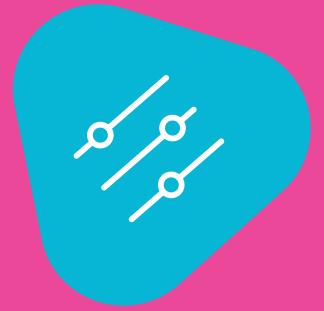


Table of Contents

- 1 Get to Know DORA
- 2 Need a Different Perspective?
Look to SPACE
 - ▶ SPACE Metrics
- 3 Find the Flow
 - ▶ Flow Items
 - ▶ Flow Metrics
- 4 General Development
Performance Metrics
- 5 The Right Metrics for the Job
- 6 How Allstacks Can Help
 - ▶ Continue the Metrics Journey

Allstacks' core value is providing visibility into problems. I can organize all of my engineering processes in one place to quickly make changes and better understand where my inefficiencies lie. I get a clear picture of where we are, so I can determine where we want to be.

John Steinmetz

VP of Data & Analytics, shiftkey



Introduction

No organization can afford to gamble when it comes to engineering efficiency.

Managers need to confidently predict development and delivery timeframes, not to mention make a case for additional resources or investments. And it's equally important for teams to get ahead of bad news because, on average, more than half of software features are delivered late, and nearly half of projects conclude over budget. In the fast-paced world of software development, "surprises" are rarely happy events, particularly for the business side.

At a time when economic headwinds continue to create corporate upheaval, organizations need to lean into what they can control, meaning it's time to stop [SWAG-ing](#) and start actually measuring engineering performance.

A comprehensive system of metrics (and dashboards) means everyone from the CEO to the finance team can see a software development project's status, performance, and trajectory concerning business goals and strategic initiatives with no guesswork or assumptions allowed. The data and the history bring to light the nuances in software development and unite the organization around shared facts and language. After all, organizations must be data-informed before they can become data-driven.

The good news is that these engineering metrics are easy to set up and maintain while being simple to use and understand. The slightly less good news is that there are many metrics to choose from in the software development space, ranging from popular “sets” of metrics to detailed stand-alone or combine-together options. It’s always best to approach engineering efficiency metrics with an open mind and an understanding that no two organizations are likely to want to measure exactly the same thing. (That’s why there are so many options to choose from!)

Here’s our take on the most important development metrics, what they can bring to an organization, and how to get the most out of them.

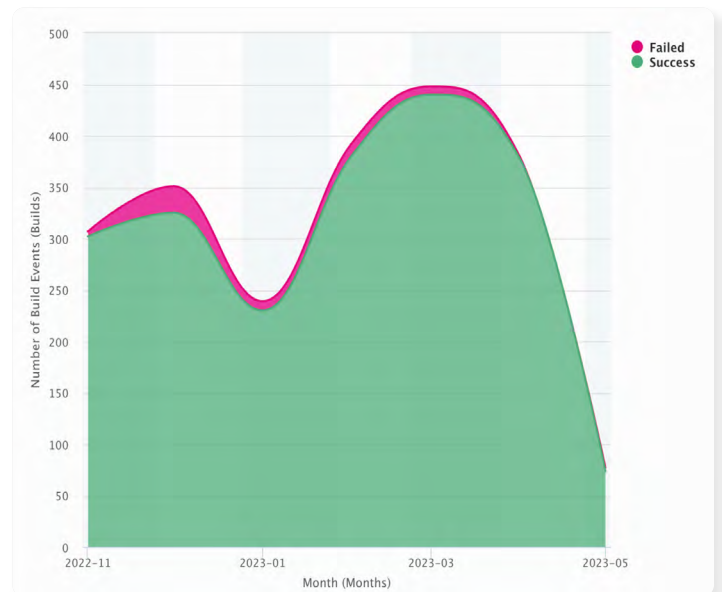


SECTION ONE

Get to Know DORA

The idea of DevOps Research Assessment, or DORA, started in 2013 when the automation company Puppet released a State of DevOps report, reportedly the first time any organization took a snapshot of the “new” methodology. Over the next few years, more surveys were conducted, more experts got involved, a company was created, and the [Accelerate](#) book was published in 2018, co-authored by three of the most notable thought leaders in the DevOps space: Nicole Forsgren, Jez Humble, and Gene Kim. Soon after that, in 2019, DORA was acquired by Google.

The [DORA metrics](#) are generally held to be the measurement gold standard for DevOps teams, and they are widely used today, often in combination with other metrics efforts. It’s important not to underestimate their importance: at a time when teams everywhere were trying to crack the DevOps code (Is it culture? Is it technology? Is it something else?), DORA metrics provided the first-ever roadmap to success as well as answers to those nagging questions (it’s culture AND technology AND process).

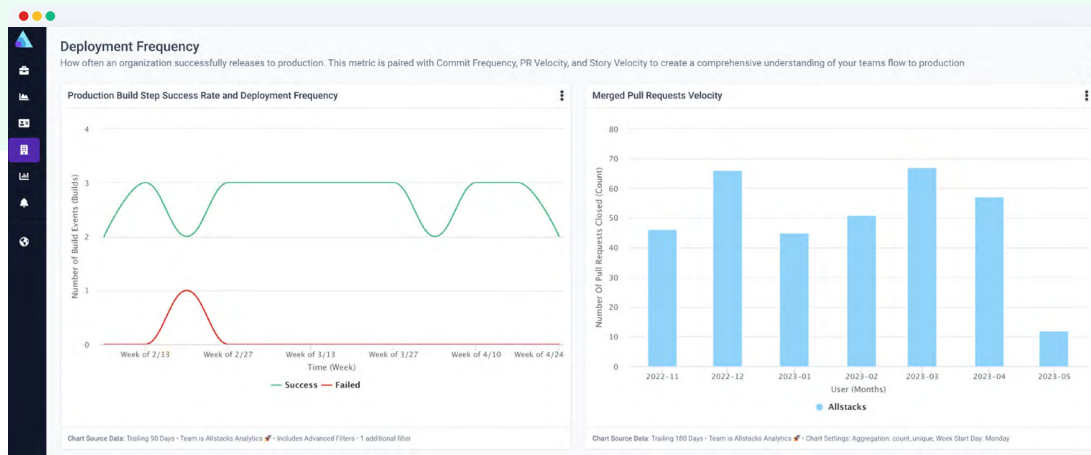


To put it another way, DORA made “doing DevOps well” a less amorphous proposition with its inception, introducing a concrete framework to measure results as companies adopted DevOps principles in swaths with hopes of delivering their software better and faster. It also gave development teams a way to evaluate their performance more defensibly: with real data and consistent measurements of success.

The fundamental premise behind DORA metrics was simple: after studying so many teams trying to implement DevOps successfully, which measurements were most useful to include? Over time the metrics have been fine-tuned, and so have the definitions of success. For many years DORA slotted teams into four categories: elite, high, medium, and low-performing.

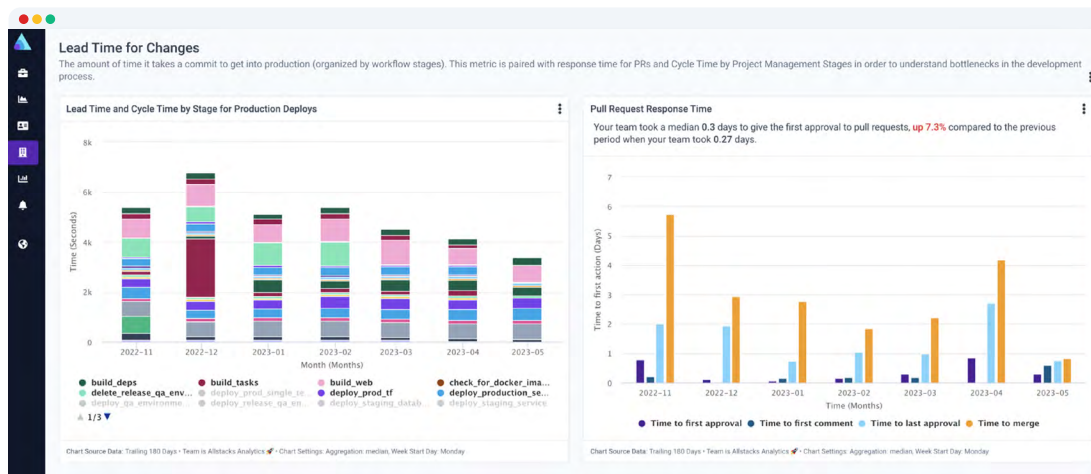
As of the most recent report ([September 2022](#)), DORA eliminated “elite” and settled on five key DevOps metrics teams should measure themselves against: **deployment frequency, lead time for changes, mean-time-to-restore, change fail rate, and reliability.**

DORA Metrics



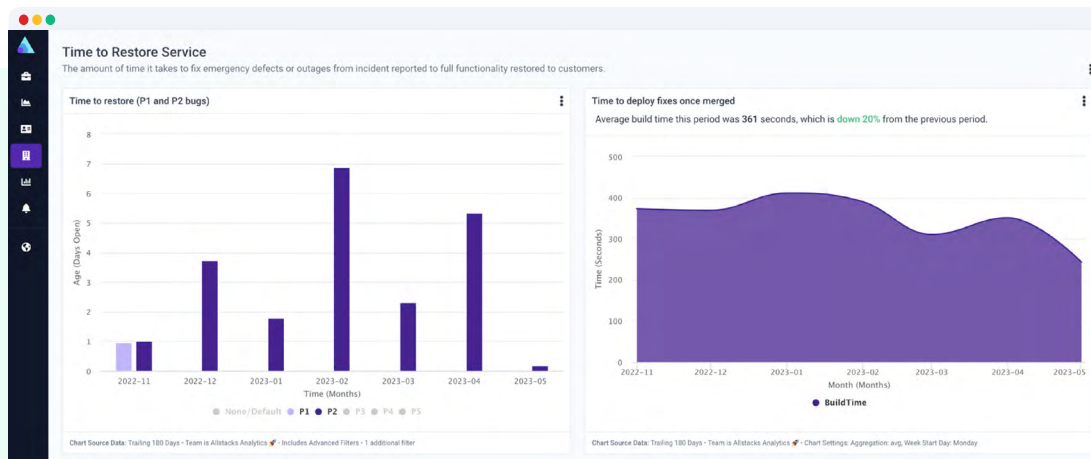
Deployment frequency refers to how often a team pushes code to end users. A high-ranking team deploys on demand (meaning multiple times a day), while a low-ranking team delivers new code between once a month and once every six months.

How often do you successfully ship to production? Allstacks ingests commit frequency, PR velocity, and story point velocity data to give you a comprehensive understanding of your flow to production and the deployment frequency at which teams release code to end users.



Lead time for changes is a metric that examines how quickly the pipeline functions. Once the code is committed, how soon will it be in production? Teams in the high category have a lead time for change between one day and one week, while those in the low category can take one to six months.

How much time does it take code changes to reach production throughout each stage of your development workflow? Allstacks pairs cycle time by project management stages and PR response time data to help you identify bottlenecks in the development process, starting from the initial code commit.



Time to restore service examines how swiftly a team can restore operations after a downtime or incident. High-performing teams are fast, taking less than a day on average. Low-performing teams take between one week and one month.

Can you confidently measure the amount of time it takes to fix emergency outages, defects, or bugs to understand the impact on your customers? Allstacks makes it possible to understand exactly how long it takes to restore full functionality to customers in the event of an incident.



Change failure rate drills down into code quality, looking to see what percentage of new code or code changes needs to be fixed after being put in production. High-performing teams have change failure rates between 0% and 15% while low-performing teams range between 46% and 60%.

Do you know the exact percentage of deployments that cause a production failure? Allstacks pairs data from successful merge rates for PRs and Issues to calculate the overall rate at which changes are unsuccessfully deployed to production and require remediation by development teams, giving teams a better understanding of deployment quality.

The newest metric, **reliability**, targets how a software service performs. Reliability looks at how available a service is, what latency rates are, the potential for scalability, and overall performance. Teams that are heavily automated use service-level objectives and/or service-level indicators, and have an incident response plan are likely to rank more highly.

When it comes to improving engineering performance, it's tough to conceive of a better place to start than with the DORA metrics. Organizations can start using DORA to evaluate speed and stability in aggregate across development teams and gauge DevOps maturity along their journey to modernize and improve software delivery. These metrics are best served in a dashboard where the entire team can see them in the context of their own DevOps efforts. Teams undergoing digital transformation, new to DevOps, and even greenfield startups can particularly benefit from the solid and sensible wisdom found within these metrics.

SECTION TWO

Need a Different Perspective? Look to SPACE

Because teams are made up of individuals, it can be useful to consider a framework that looks at the people within the process, and in the software development arena, that option is SPACE. Unlike the concrete DORA metrics, SPACE is more of an umbrella perspective organizations can adopt to help them get a handle on the very opaque concept of developer productivity. SPACE is a way to think about productivity metrics, with lots of suggestions and cautions, rather than a black or white measurement overlay.

SPACE is an acronym that stands for: **satisfaction** and **well-being**, **performance**, **activity**, **communication** and **collaboration**, and **efficiency** and **flow**. Developed by Nicole Forsgren, a Vice President at GitHub, along with a number of DevOps experts and academics, SPACE was more or less born out of the remote work explosion caused by Covid-19. And because SPACE was created by those literally working in the developer field, it provides a thoughtful and structured way to look at developer productivity *and* job satisfaction—it's meant to be tailored or trimmed down to fit individual team or organizational needs. Forsgren goes so far as to caution management not to overwhelm developers with too many productivity measurement metrics she suggests choosing at least three and ensuring each looks at a different facet of productivity.



The Five SPACE Metrics

Satisfaction & Well-Being

How fulfilled, happy, and healthy one is

- Investment balance (KTLO)
- Team health check
- Employee satisfaction (eNPS)

Performance

Outcome of a process

- Investment balance (building new stuff)
- Review rate
- Change failure rate (DORA)
- Customer satisfaction (NPS)
- Reliability

Activity

The count of actions or outputs

- Issues completed
- Pull requests merged
- Deployment frequency (DORA)

Communication & Collaboration

How people talk and work together

- Team health check
- Pull request review time
- Number of collaborators per story
- Cross-team review times

Efficiency & Flow

Minimal work delays or interruptions

- Team health check
- Investment balance (productivity)
- Change lead time (DORA)
- Time to recovery (DORA)
- Issue cycle time

1

Satisfaction and Well-being

Are the tools and tasks resulting in developer job satisfaction? Is a developer happy and healthy in life? These are the questions this metric aims to uncover. Forsgren makes the case that satisfaction and productivity are inextricably linked, so it behooves organizations to dive into this area. The best way to use this method is through surveys—ask employees about the state of the culture, whether the right tools are in place for the job and the possibility of burnout.

2

Performance

File this metric under the “it’s complicated” flag. Performance is perhaps the most difficult area to pin down simply because of software development’s collaborative and complex nature. A dev might write a lot of code, but it could be error filled, while a feature created by a team might be popular with customers but not make the company any money. Forsgren’s answer to this problem is for organizations to look at “outcomes” instead of “output.” With that in mind, her suggested metrics revolve around (code or service) quality and (customer or business) impact.

3

Activity

Broadly, this is a suggestion to look at what’s happening during a day, a sprint, or an iteration, but it comes covered in caveats that, like performance, can be difficult to measure objectively. Forsgren suggests looking at team activity through three lenses: design and coding, CI/CD, and operational activity. To drill down into design and coding, ask about pull or merge requests, code reviews, or commits. Break down continuous integration and continuous deployment into the number of builds, tests, deployments, or the usage level of the infrastructure.

4

Communication and Collaboration

How effectively does a team work together? Communication and collaboration were so-called “soft skills” not taken particularly seriously in the early days of software development. Still, they have achieved a more prominent role of late with the advent of remote work and highly distributed teams. The best way to uncover the state of communication and collaboration is by looking at various factors, including documentation, code reviews (and frequency), and onboarding.

5

Efficiency and Flow

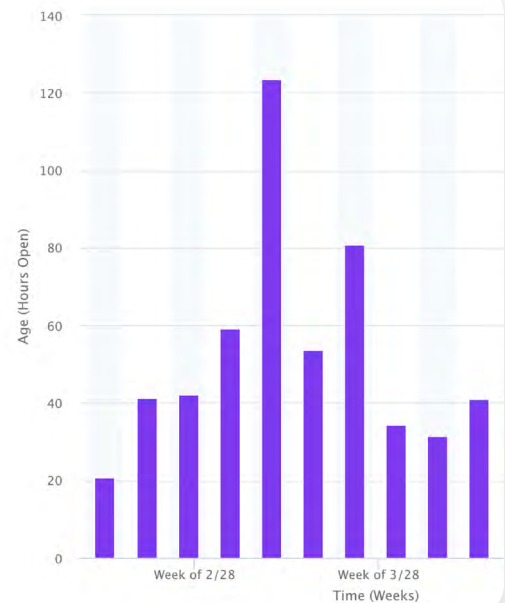
Though grouped together, Forsgren makes it clear efficiency is something baked into all of the SPACE metrics, while flow refers to that ideal state when an individual (or team) is humming along with minimal interruptions or distractions. Organizations can ask developers directly about the flow state—how often is an efficient, ideal state achieved with minimal disruption to planned, value add work?

SPACE Metrics

Other metrics to consider are handoffs, wait times, and interruptions, because these uncover parts of the development workflow where unnecessary or extra context switching negatively impacts developer efficiency.

Teams of any size and in any industry can take advantage of SPACE to increase engineering efficiency, but organizations with sizable development teams might find the metrics framework particularly valuable. With its focus on flexibility, one-on-one communication, and outcomes, SPACE is a foundational starting point for any organizational conversation around developer performance, efficiency, or productivity.

Before moving on, it's also worth noting that although SPACE and DORA focus on separate things, they are related. In the words of Nicole Forsgren, creator of both, "DORA is an implementation of SPACE." They coexist because the SPACE framework defines a paradigm with specific dimensions to assess and understand productivity, whereas DORA can be an application of SPACE to measure performance.



Find the Flow



Teams using DORA or SPACE, or even both, still may find a hole in their efficiency measurement strategy: Is the development team actually working on projects that will bring the most value to the business? Traditionally mapping business value onto software development has been difficult to do as neither side speaks the same language, which can easily lead to a lack of shared priorities and understanding.

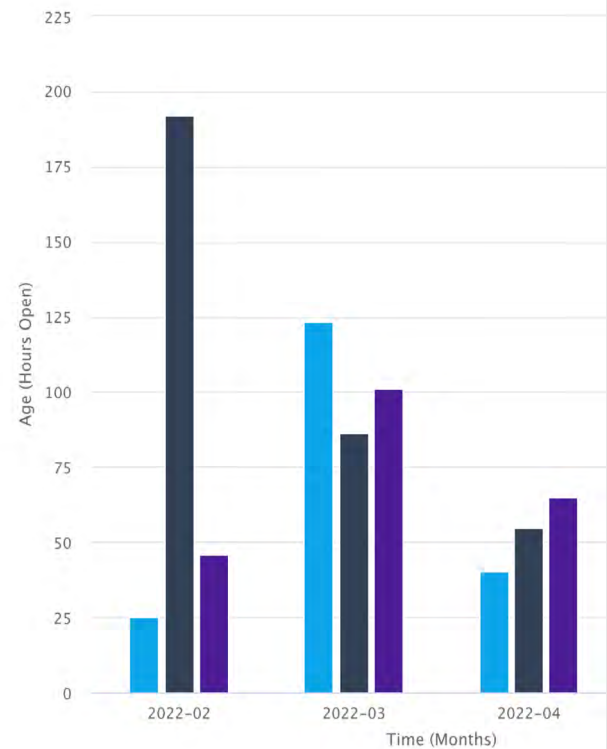
[Flow metrics](#) bring the two sides together, ideally with minimal pain and maximum collaboration. The Flow Framework is the brainchild of [Dr. Mik Kersten](#), who in 2018 got the idea to map four software flow items (representing the key stages of development) against six measurement metrics. If implemented correctly, both the development and the business side will have a high-level view of the process and the business value being created.

Flow metrics can revolutionize how the business of software development happens; aligned teams will be able to track progress and forecast risks and outcomes using a shared set of data and terminology. A successful flow metrics implementation will remove all the second-guessing about product features, UX, customer satisfaction and even investments in teams and infrastructure and replace it with civil, informed discussions of priorities... at least in theory.

Flow Items

Everyone involved in development likely knows what “flow” is—that state where things just happen cleanly and effortlessly. The flow items are Dr. Kersten’s way of breaking up the rather complex software development process into four parts.

Features are what is being built, and this metric is tied completely to code creation. It is also closely related to business priorities. **Defects** is an umbrella term referring to debugging and of course the time it takes to find and fix problems. **Risk** is the metric that reflects the precarious nature of code development, i.e., it can be at risk of security breaches or not being compliant. And finally **debt**, sometimes known by the dev term technical debt, is a tip of the hat to flow, as debt stands in the way of forward momentum in software development.



Flow Metrics

Velocity



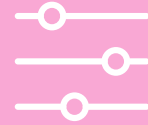
How much value do we deliver?

Speed



How fast do we deliver value?

Load



Is demand overwhelming capacity?

Efficiency



Where are the bottlenecks?

Distribution



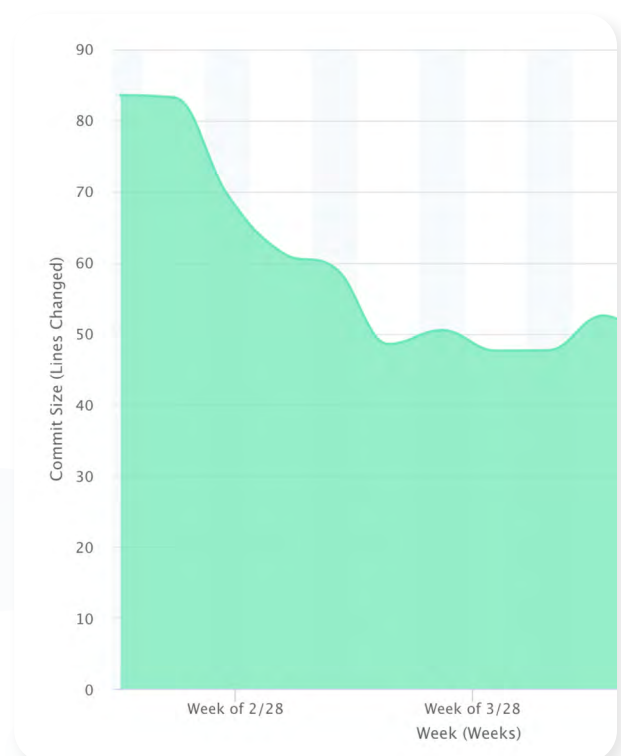
Are priorities in line with business objectives?

Predictability

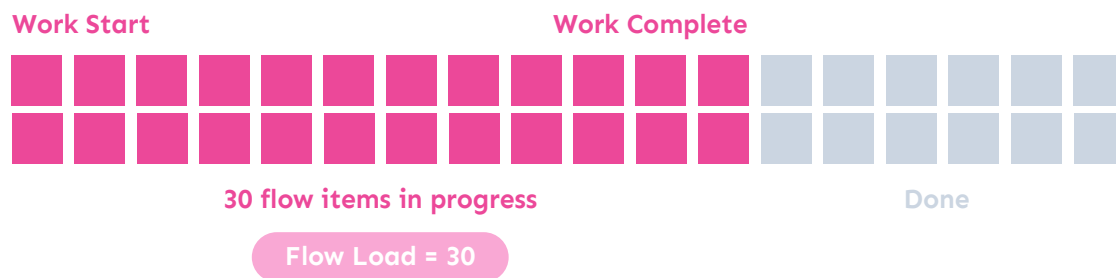


Are you prepared for any last-minute surprises?

The OG flow metrics from 2018 contained only five metrics—velocity, time, efficiency, load and distribution. But [the Scaled Agile Framework \(SAFe\)](#) has suggested a sixth metric—predictability—and that has become widely accepted as part of the flow metric framework.



Flow velocity is a relatively straightforward metric that looks to answer the question: How much work is happening during a specific timeframe? Teams can unpack the velocity rate by looking at the development backlog or throughput. The goal is for velocity to at least stay the same and ideally increase; a decrease in velocity likely warrants investigation. Flow velocity and another flow metric, flow time, are colloquially known as the “money makers” because they’re two quick ways to see if the software development team is actually creating value.

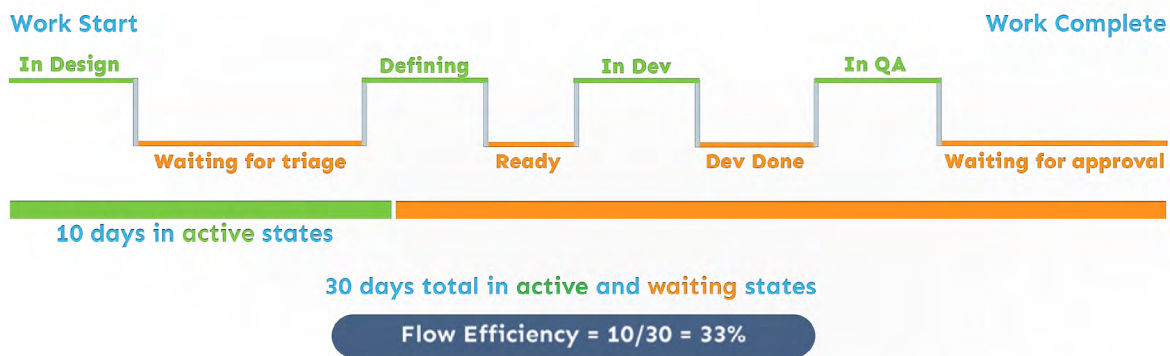


Flow time asks the next logical question after flow velocity: How much time is it taking for teams to produce a flow item? Flow time is measured from beginning to end, even if no work is happening, and this is definitely a metric teams would like to see decreasing. Flow time is also very useful for future planning; teams can look at the data history to be able to forecast how long a project *should* take.

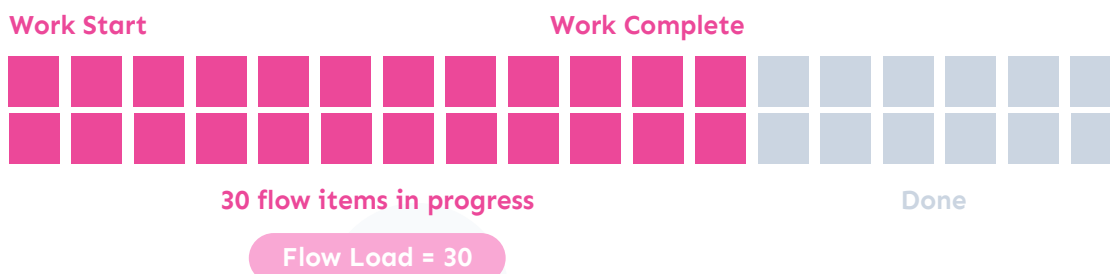


Source: [Flow Framework](#)

Work is happening at a brisk pace, but is it efficient? This is always one of the most sought-after answers while also being one of the most difficult to get to. **Flow efficiency** tries to tackle this dilemma with data by dividing activity (flow velocity) by flow time. And unlike other efforts to judge efficiency, this metric looks at the entire software development lifecycle from ideation to delivery and takes into account all delays, inactivity and other bottlenecks. The flow efficiency metric will save the day for teams struggling to find logjams.



Organizations want productive development teams, but they also don't want burnout. The flow load metric is a way teams can help define that tricky balance. **Flow load** examines the number of flow items or works-in-progress to see exactly what's happening at each stage of development. Too many items in the works might mean a team is exhausted or overwhelmed while too few might mean it's time to rejigger in order to get the most value out of the process. This is definitely a "your mileage may vary" metric though, because everything from organization size to developer experience level can affect this metric.

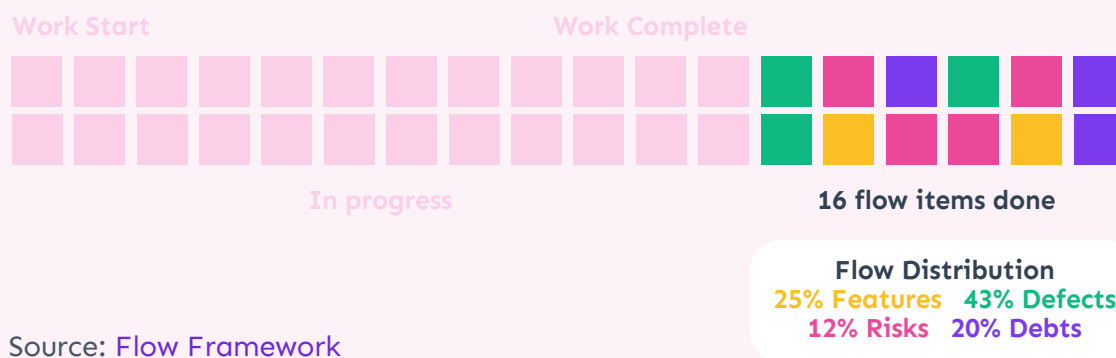


Source: [Flow Framework](#)

Teams can turn to the flow distribution metric to answer the universal developer lament of “What should we be working on?”

Flow distribution can let teams actually see what makes up their workload and where there is too much or not enough. This information can help decide how much time should be spent on UX versus technical debt, as just one example.

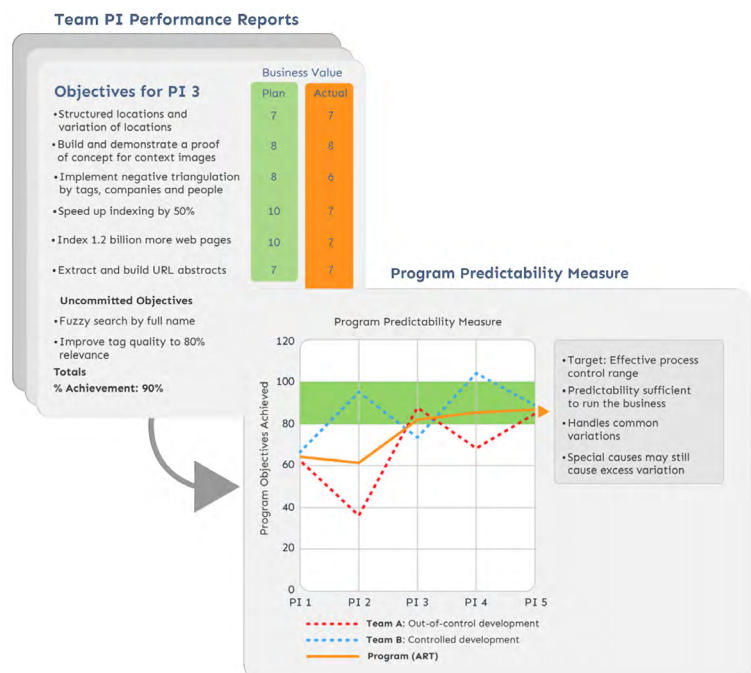
And finally, flow predictability is all about consistency throughout all the metrics. Managers can know at a glance which parts of the development process are consistent and which are struggling, and that information can inform everything from priority setting to resource allocation.



Source: [Flow Framework](#)

Flow metrics are an option for any development team using any methodology and in any and all industries, and they play very nicely with other metrics frameworks including DORA and SPACE. However, they are nested metrics, meaning management should be prepared to implement them all to get the most accurate picture of the software development flow.

Source: [Scaled Agile Framework](#)



SECTION FOUR

General Development Performance Metrics



As we mentioned earlier, there is no dearth of metrics in the software development space. A number of measurements go beyond DORA, SPACE and flow metrics, and that can certainly be used alongside them either regularly or in specific circumstances. Not every metric is meant for every team. Here's a look at some of the more popular performance metrics that could improve engineering efficiency:

- **Days coding:** How many days a week is a team coding? This metric looks at productivity but is also useful for planning purposes.
- **Planned versus unplanned issues:** This measurement should be of interest to engineering management wondering how developers are spending their days. Too much time spent on “unplanned” issues might indicate a need for better team organization or planning.
- **Cycle time by issue type:** What issues are developers working on, and which take the longest to complete? Are bugs taking longer than stories or tasks? Are teams spending a lot of time on miscellaneous issues? This metric can answer those questions.
- **Types of code:** Teams produce several different types of code, whether it's legacy refactoring, new work, or comments. This metric tracks what's being created to help distribute workloads more evenly.

General Development Performance Metrics

- **Commit volume:** Sometimes known as Lines of Code (LOC), commit volume is a way to look at developer productivity trends broadly. Volume can vary over a week or a month, so ideally, this metric is looked at with an eye to evaluating an individual developer's output over time and not to compare developers to one another.
- **Commit size:** Size matters when it comes to commits because the larger the commit, the greater the chance it could contain a security risk that might get overlooked. Smaller commits are overall safer and easier to scan for potential security issues.
- **Commit frequency:** Over a given period of time, how often is code committed? As with many of these metrics, context is key—a more junior developer can be working hard and still commit less code than someone with more experience.
- **Work In Progress (WIP):** Track how many projects teams and/or individuals have moved out of the backlog, and see how much more effort will be required.
- **Number of releases:** How often an engineering team releases new code is a common productivity metric.
- **Technical debt:** What problem code is hanging around and hasn't yet been dealt with? Technical debt is a reliable way to look at development efficiency and judge the health of existing tools, processes, and infrastructure.
- **Mean time to detect:** Bugs happen, so how long does it take to detect them? This metric is useful from both an efficiency perspective and can give valuable input about processes and infrastructure.
- **Mean time between failures:** This metric looks at the frequency of code "failures"—obviously, the greater the amount of time, the better.

The Right Metrics for the Job

Metrics help an organization discover what it doesn't know, and the data uncovered may be startling at first, but will ultimately be liberating. It all starts with visibility, transparency, and intentionality when setting expectations. Perfection can't get in the way of progress. Only after establishing a baseline can meaningful insights and trends be extracted from data to drive healthy cultural change and foster productive conversations about how to improve. Teams can't fix what they don't know, and no one can afford blissful ignorance in today's results-oriented world.

But companies also can't afford burned-out developers or high levels of employee churn, and that's actually the biggest surprise benefit to implementing measurement metrics: everyone will be able to see what's working, what's inefficient, what's missing, and where and how to move forward with investments, personnel, and technology. Metrics power productivity and [the connection between developer happiness and productivity](#) has been proven.

When developers are both productive and happy, they're more confident in the value they create for the business internally and understand the impact of that value for customers externally. However, without metrics, it's impossible to quantify success and qualify what needs to be done to improve.

The Right Metrics for the Job

For example, there's immense value in changing a narrative from, "This feature or product took way longer to ship than expected, what's going on?" without supporting data to back it up, compared to, "Historically, PR cycle time and code review are our biggest 'wait states' throughout our development workflow, accounting for a significant proportion of time where work stalls out—code review should have taken two days for this release, but it took five days because there were no comments in the PR to help move through code review efficiently." One approach paints a picture of uncertainty and doubt, and the other identifies where the problem lies and equips teams with the required data to improve.

And finally, there is much talk about the idea of the 10x team, that rarest of groups that is efficiently collaborating, in a state of flow, totally engaged in the work, and getting the job done. To get to 10x, or 5x, or to just better engineering efficiency tomorrow than today, start with metrics.

SECTION ONE

How Allstacks Can Help



Regardless of where your organization is on the journey to transform and improve software delivery, engineering metrics can't be a black box and data can't be left in the dark. DORA, SPACE, and FLOW are all useful frameworks to measure efficiency, productivity, or performance with unique aspects to each approach that shine a light on what “good” looks like. They produce compelling, powerful metrics—but metrics need to act as a catalyst for actions to follow and include the right context to be most valuable. When metrics open up lines of communication and collaboration across teams, and lay a foundation to ask the right questions, data becomes an agent of positive change.

The first step is to “turn the lights on” in regards to data and metrics. Get visibility into what work is being delivered, when to expect it to be done, and gain an understanding of how to improve based on actionable intelligence, not opinion or guesswork. That's where [Allstacks](#) comes into play.

Our Value Stream Intelligence platform helps companies of all shapes and sizes align business outcomes to engineering workflows by ingesting data from tools across the entire SDLC so that teams can measure, grow, and optimize how they deliver their software—together. Create metrics your way, customize visual dashboards, and share learnings in order to make data-driven decisions that make software delivery more efficient and predictable.

If you're interested in giving a voice back to engineering at your organization, [let's talk and see how we can help.](#)

Continue the Metrics Journey

How can Allstacks support you?

How metrics lead the way through a retailer's "everything transformation"

Learn more about commit volume

Why looking at commit size can make for more secure code

How does your team's engineering efficiency compare?

**Book a demo today or
try Allstacks for free.**