EXPERT GUIDE

The Engineering Leader's Guide to Accelerating Developer Productivity

LinearB

DEVELOPER PRODUCTIVITY

Summary

Developer Productivity refers to the effectiveness and efficiency with which software developers produce high-quality code and complete projects. It encompasses various aspects, including:

- **Code Quality:** Writing clean, maintainable, and bug-free code.
- **Speed:** The pace at which developers complete tasks and deliver features.
- **Efficiency:** How well developers use tools, resources, and time to achieve their goals.
- **Problem-Solving:** The ability to effectively tackle and resolve technical challenges.
- **Collaboration:** How well developers work with others, including communication and teamwork.
- **Adherence to Best Practices:** Following industry standards and practices to ensure robust and scalable solutions.

Improving developer productivity often involves optimizing workflows, providing the right tools, reducing distractions, and fostering a supportive work environment.

However, the notion of measuring–much less accelerating–developer productivity is often met with healthy skepticism if not outright hostility. Common refrains range from “The simple act of measuring something means it’s not a good measure anymore” to “Development work is too creative and complex to measure and quantify” to “This is going to kill our culture.”

LinearB

DEVELOPER PRODUCTIVITY

But all the resistance and objection in the world–however valid–doesn't change the fact that organizational leaders, most notably the C-Suite, are under increased pressure to improve efficiency and value creation. This is especially true in the world of flat and/or shrinking budgets. The Pragmatic Engineer's Gergely Orosz, put it well: "The executive reasoning goes: if other groups [like sales, marketing, and CS] can measure performance, it's absurd that engineering cannot."

Clearly, this determination to measure and optimize developer productivity is here to stay and will be implemented industry-wide. What's not yet set in stone is how best to go about it–and this is one of the primary challenges with developer productivity initiatives.

What's covered in the pages ahead:

- **Proving Productivity–A Profound Project:** why measuring and working to optimize developer productivity is so difficult but leads to the promised land.
- **Productivity Predictors:** the metrics and KPIs that illustrate productivity and provide you with a baseline of how to improve.
- **Accelerating Developer Productivity:** a step-by-step overview of the process of measuring productivity and using this data to set goals, inject automation, and inform conversations with your team.

Table of Contents

LinearB

Proving Productivity

It’s unlikely that you’ll meet an engineering leader who doesn’t want to measure and optimize their team’s productivity. Everyone wants to quantify and improve their efficiency and impact on the organization. The reality is that this is easier said than done because of the significant hurdles involved. However, those driving increased engineering productivity will enjoy a significant competitive advantage.

The Challenges and Benefits of Developer Productivity Improvement Initiatives

The ability to collect data from leading DevOps tools and support for common engineering metrics frameworks is a must-have requirement for Developer Productivity improvement initiatives. This is a typical starting point for organizations building a data practice. Teams get enhanced visibility into the overall software development life cycle with insights generally focused on developer productivity use cases. DORA specifically provides a high-level view of velocity and quality.

Anything worth having doesn’t come easily–improvements to developer productivity are no exception. Whatever your engineering organization looks like, odds are you’re dealing with at least one, if not all, of the following hurdles. However, when you successfully clear those hurdles, the benefits to your engineering organization and the business cannot be overstated.

Let’s explore these challenges and benefits in more detail.

PROVING PRODUCTIVITY

Data and Insight

The desire for lots of data is a complex issue. Wanting more data is a solid and rational position, as within data lies the insights that when acted upon can help you:

- Make the best decisions possible in any given situation
- Dramatically increase developer productivity

However, for many engineering organizations an overabundance of data exists--often coming in different formats from different sources and at different levels. Generating meaningful, actionable insight from this data can become a full-time job, which takes too much time and resources.

When teams break down data silos and see project management, source code management, and engineering resourcing data in one place–improving developer productivity becomes a much easier task.

End-to-end visibility–from how work flows through the pipeline to how engineering investments and project resourcing impact overall spend and budgets–enables every part of your organization to make better decisions and optimize ways of working. Correlating these disparate data points in a single platform reveals a larger story and provides actionable insight.

Everyone from the team leads all the way up to the CTO will improve processes if they have unified visibility into what matters most to them. And it shouldn’t require 10 different dashboards in 10 different tools to get it.

"...it shouldn’t require 10 different dashboards in 10 different tools to get it."

PROVING PRODUCTIVITY

Software Delivery Processes

Developers aren’t machines. Ascribing numbers to development work is likely to be met with resistance because ideation, scoping, planning, and other intangibles are part of the development process. Simply put, there is nuance to this type of work, and evaluating productivity based on output like lines of code or deployments to production per week misses important context. This leads to the industry often measuring developer productivity incorrectly–only measuring effort and output, and omitting planning and execution metrics.

Because of the use of incorrect metrics like lines of code or correct metrics being used improperly, such as attempting to cut out conversations because you have data now, it can be difficult to come up with an agreed-upon measure for developer productivity. A great example of this state of affairs is the ongoing debate concerning the McKinsey report on the subject.

There is also a significant lack of standardization. Because of the variance in tools and practices across different organizations, as well as the common practice of using “wrong” metrics or applying the “right” ones incorrectly, there’s a high risk of negative outcomes.

When nuance is accounted for and metrics are applied correctly, data enhances rather than replaces conversations. Most importantly, when insights are acted upon, teams will experience greater efficiency, improved business outcomes, higher developer satisfaction, and continuous improvement across all aspects of their engineering practice.

LinearB

PROVING PRODUCTIVITY

Change Management

You've likely experienced significant change within your engineering organization. Whether through mergers and acquisitions, RIFs, tooling, infrastructure, or new policies--such as security or compliance mandates-- some changes lead to productivity gains.

It often feels like the only constant in engineering is change– projects are re-prioritized, new tools are adopted, and teams merge. That's why the ability to quickly adapt to change, anticipate any potential impact on productivity, and maintain focus on the real work of delivering new value is so highly prized. Effectively cultivating this behavior requires deep visibility into:

- Processes
- Team dynamics
- Project resourcing and budgets
- Developer health and experience

With unified, correlated visibility into these aspects, you can now see whether change has shifted focus away from establishing smooth operational cadences and working on value-creating projects to simply adapting to the new way of doing things.

"The secret of change is to focus all of your energy not on fighting the old, but on building the new..."

SOCRATES

LinearB

PROVING PRODUCTIVITY

Change Management continued

This visibility enables you to measure the productivity impact of change, report it to stakeholders around the business, and create an effective strategy for mitigating negative outcomes or leaning into any gains you've observed. This helps you to:

- Forecast predictable outcomes
- Keep promises made to the business
- Maintain a high level of confidence in engineering
- Master change management

BONUS BENEFIT

Developer satisfaction

The best way to ensure talent acquisition and retention is to provide a good developer experience. Productivity is among the best satisfaction indicators, as merging developers are happy developers. If you provide a supportive, collaborative environment that uses data to optimize processes and inform conversations, you will see an uptick in productivity, team member engagement, and longevity (which further spurs productivity).

These benefits, when realized together, will spur higher efficiency, a superlative developer experience, and strategic alignment with business goals–all of which contribute to higher developer productivity.

Measuring Developer Productivity

When we as an industry have too often been focused on incorrect metrics and KPIs, what are are the right ones?

Indicators of Developer Productivity

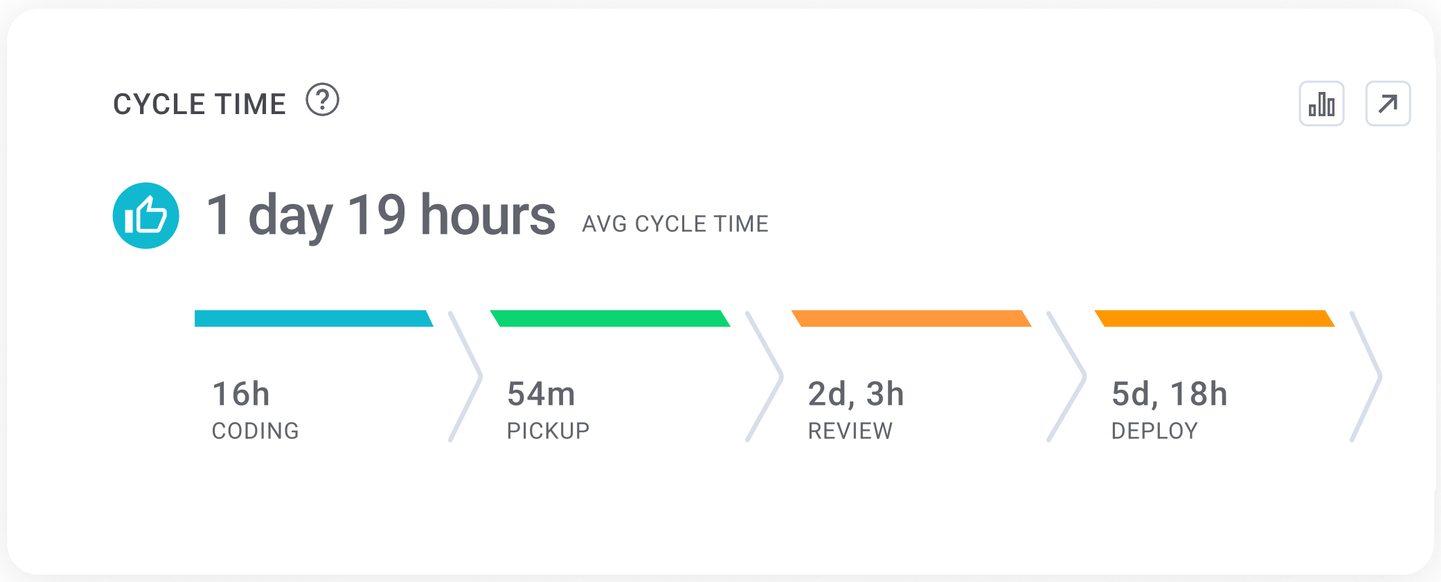
Developer productivity is a multi-faceted concept that encompasses three key areas: efficiency, effectiveness, and experience.

Each of these aspects contributes to a developer's overall ability to produce high-quality work in alignment with business goals while maintaining a healthy work-life balance and continuous growth.

Efficiency

Efficiency in development refers to the ability to complete tasks quickly and with minimal waste. This includes:

- **Cycle Time:** The time it takes to get a piece of code from pull request to production. This is one of the best indicators of development team efficiency, as it measures the entire PR process–a.k.a the most painful part of software development.
- **PR Maturity:** Minimizing the back-and-forth in pull requests to ensure quick and decisive reviews, leading to faster integration of code changes.

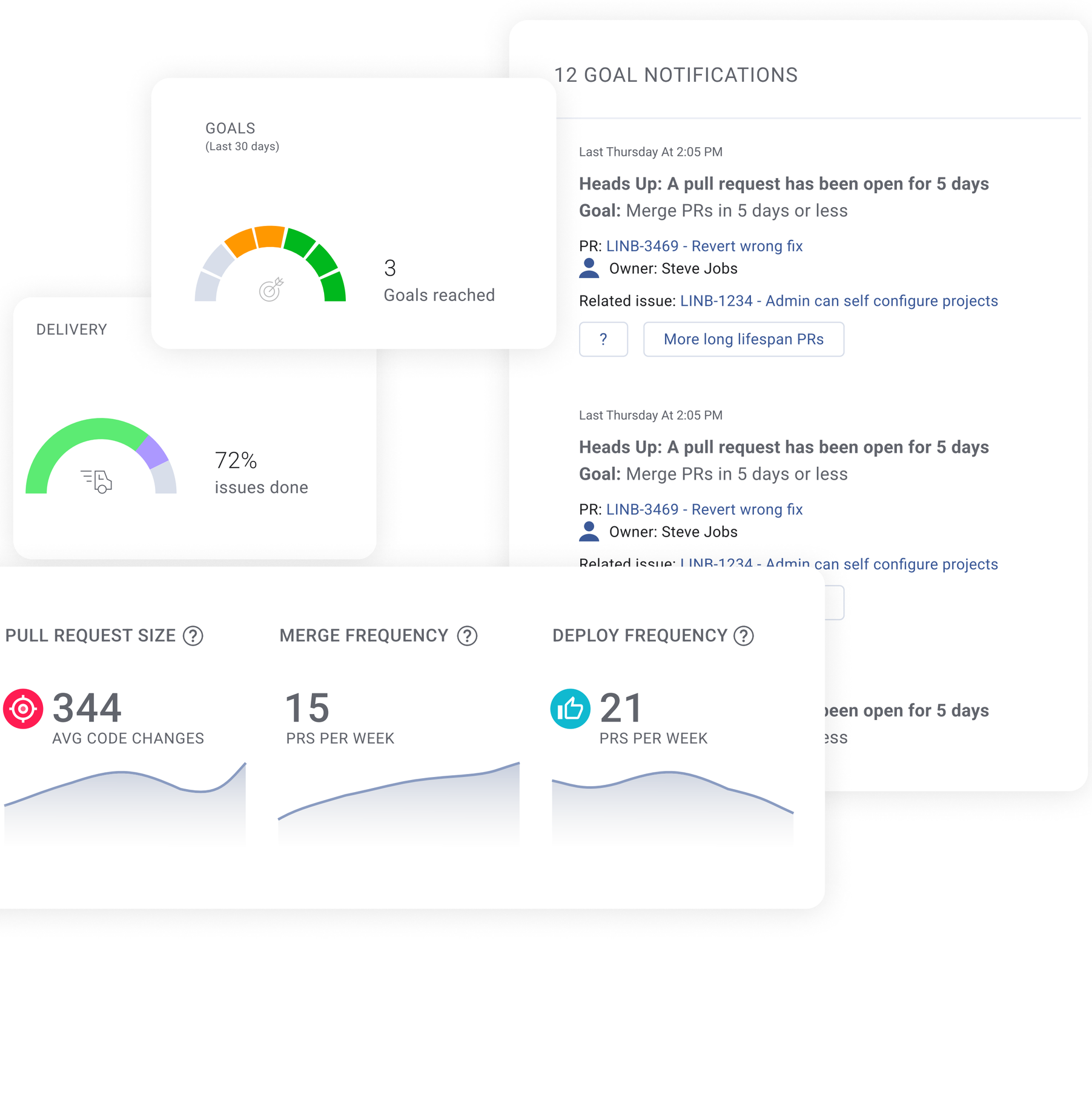


LinearB

MEASURING DEVELOPER PRODUCTIVITY

Efficiency continued

- **PR Size:** One of the best leading indicators of overall efficiency. Small PRs mean faster pickup time, reviews, and cycles–translating to faster time to value.
- **Code Churn and Rework:** Reducing the need for frequent code changes and revisions by getting it right the first time.
- **Stability (CFR and MTTR):** These are cyclical metrics. High operational efficiency and PR maturity will drive improved quality, which in turn will drive better stability. This improved stability will enable higher operational efficiency as teams will spend less time addressing bugs and outages.
- **Cognitive Load:** Avoiding multitasking, high WIP counts, and attention splitting to maintain focus and reduce cognitive overhead.
- **Task Completion:** Striving to complete tasks quickly and move on to the next task efficiently, which is considered the ideal state.

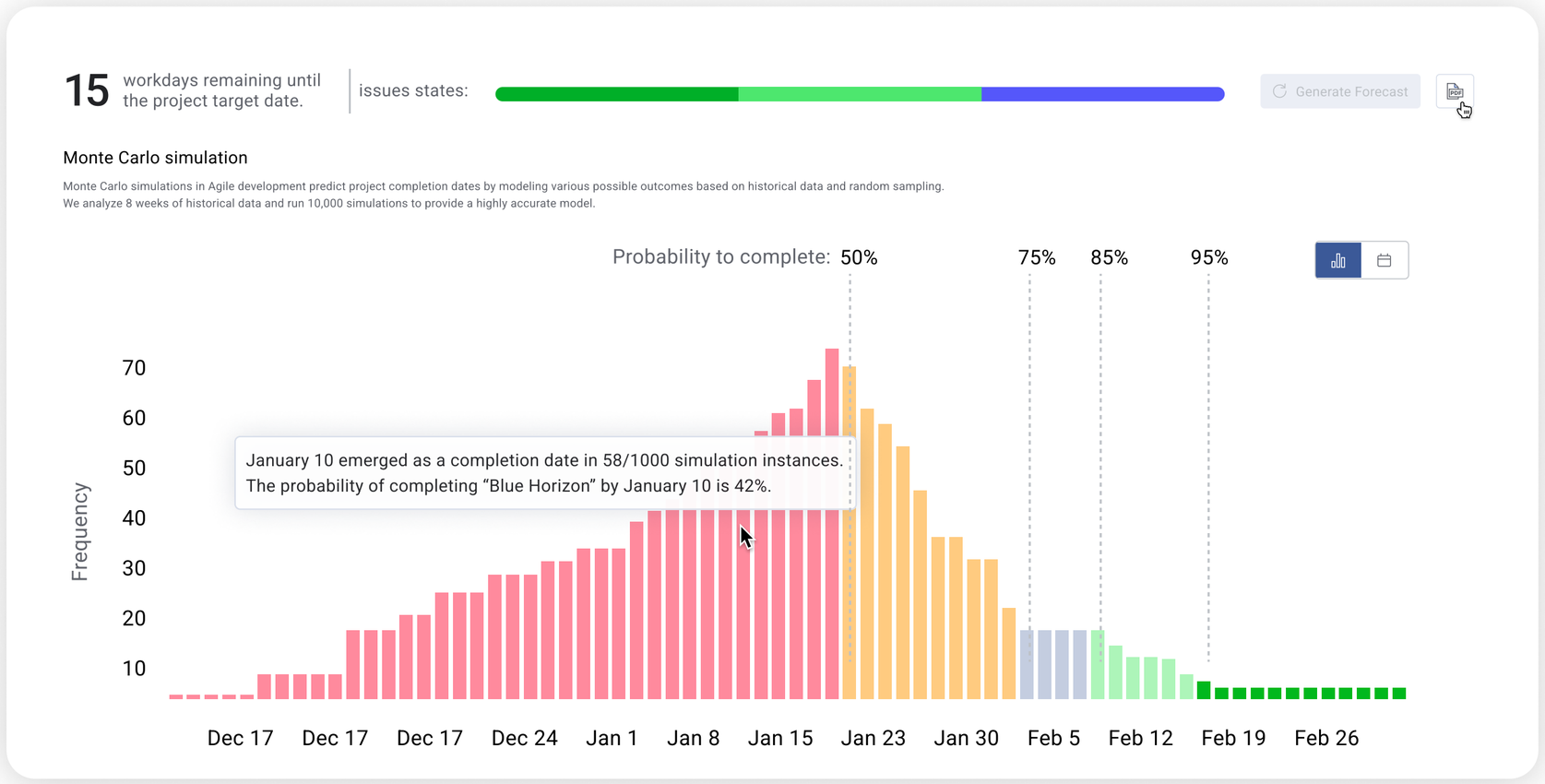


MEASURING DEVELOPER PRODUCTIVITY

Effectiveness

Effectiveness involves doing the right tasks that align with business goals and contribute meaningfully to the team and organization. Effectiveness also has a profound impact on delivery timelines and project forecasting. This includes:

- **Alignment with Business Goals:** Ensuring that tasks and projects contribute directly to business objectives and strategic goals.
- **Resource Allocation:** Assigning the right people and the correct amount of people to projects.
- **Accuracy Scores:** Focusing on scoped work (Planning Accuracy) while maintaining operation flexibility to adapt to the regular ebb and flow of engineering organizations (Capacity Accuracy).
- **Meaningful Contributions:** Making significant contributions to one's business unit or area of ownership, ensuring that work has a tangible impact–this is informed by both resource allocation and investment strategy.
- **Consistency:** While predictability and keeping promises to the business is important, so is operational consistency. Teams should strive for consistent and steadily improving accuracy scores as well as cycle time to ensure accurate timeline forecasts and kept promises.
- **Balancing New Value and Maintenance:** Striking a balance between creating new features and maintaining existing systems (keeping the lights on).

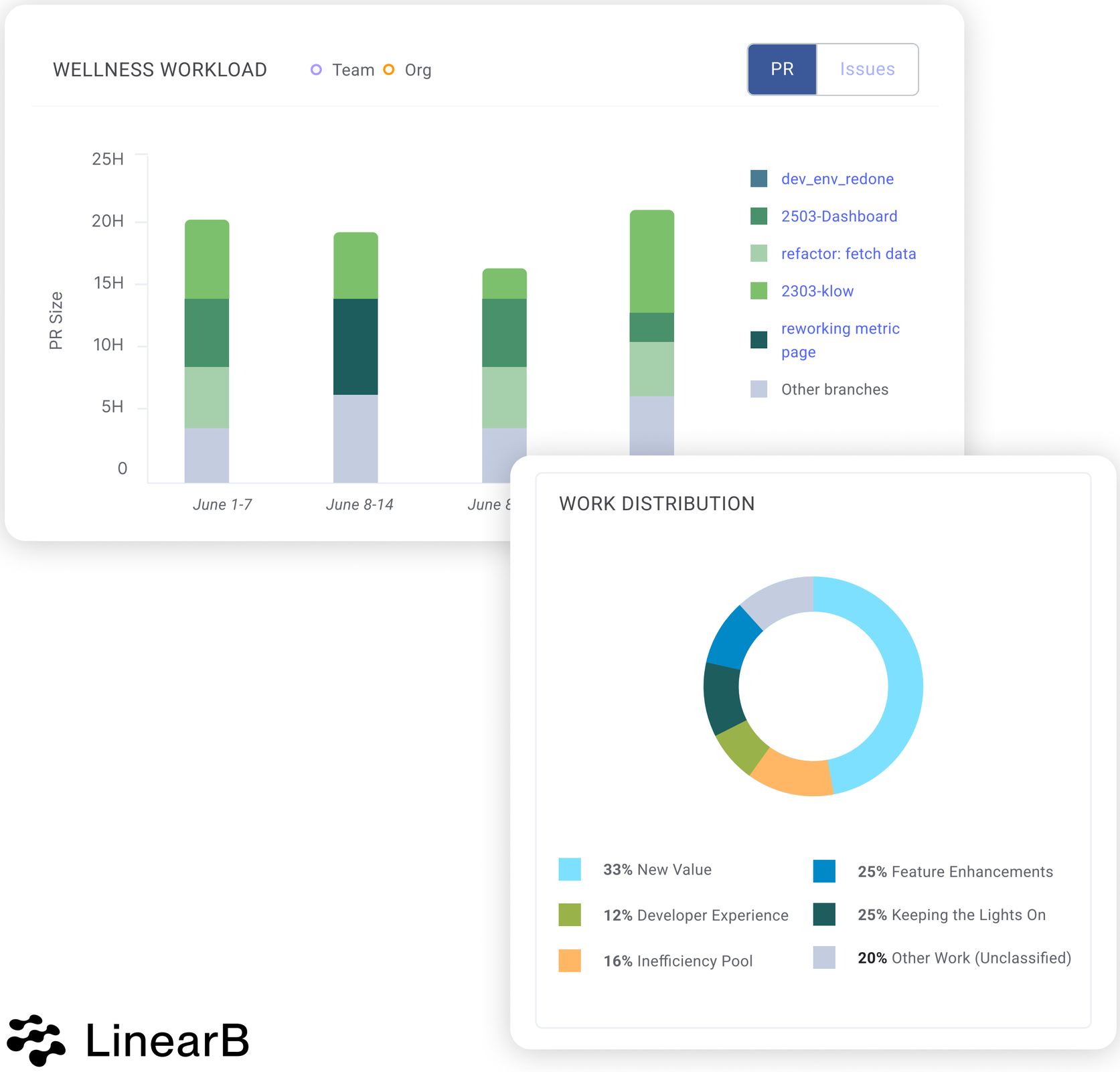


MEASURING DEVELOPER PRODUCTIVITY

Experience

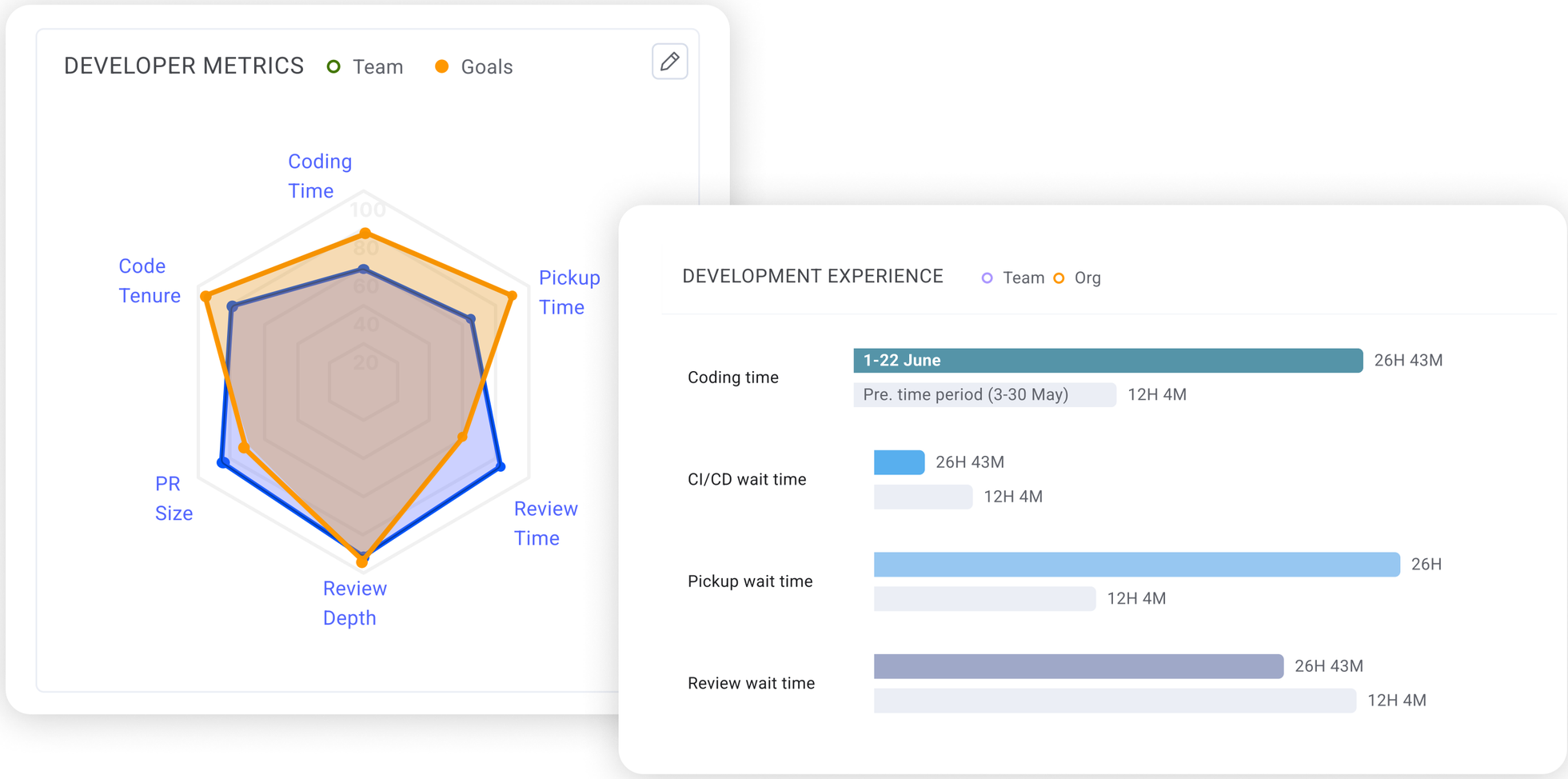
Experience encompasses the morale and engagement of developers, directly impacting their productivity and overall job satisfaction. Key aspects include:

- **Idle or Wait Time:** Otherwise known as developer toil, this metrics collection shows the time between the various phases of the pull request process. Idle time is the killer of productivity as it splits focus, creates a high cognitive load, and drives further slowdowns in the PR process.
- **Meeting Team Goals:** When bottlenecks and inefficiencies are discovered (as informed by industry benchmarks), effective teams set data-backed goals against them. They also leverage automated workflows. Regularly attaining these goals indicates both developer effectiveness and a good developer experience.
- **Task Balance:** Ensuring a healthy mix of tasks, including both bug fixes and engaging new challenges, to keep developers motivated and interested.



LinearB

MEASURING DEVELOPER PRODUCTIVITY



- **Skill Expansion:** Encouraging developers to explore new parts of the codebase and expand their skill sets, promoting continuous learning and growth.
- **Knowledge Expansion:** Continuously expanding knowledge of the codebase and learning new programming languages and technologies.
- **Coding vs. Code Review:** Balancing time spent on writing code with reviewing and providing meaningful feedback on others' code to maintain high code quality standards.
- **Unplanned and Added Work:** Unplanned work is a normal part of any engineering organization. A healthy amount of planning and scoping flexibility is good, but it needs to be addressed when it begins to impact the delivery of roadmap items promised to the business.
- **WIP:** Managing work-in-progress (WIP) to avoid overloading developers, helping them maintain focus and avoid burnout.

Organizations can significantly enhance developer productivity by understanding and optimizing these aspects of efficiency, effectiveness, and experience, leading to better outcomes for both the developers and the business.

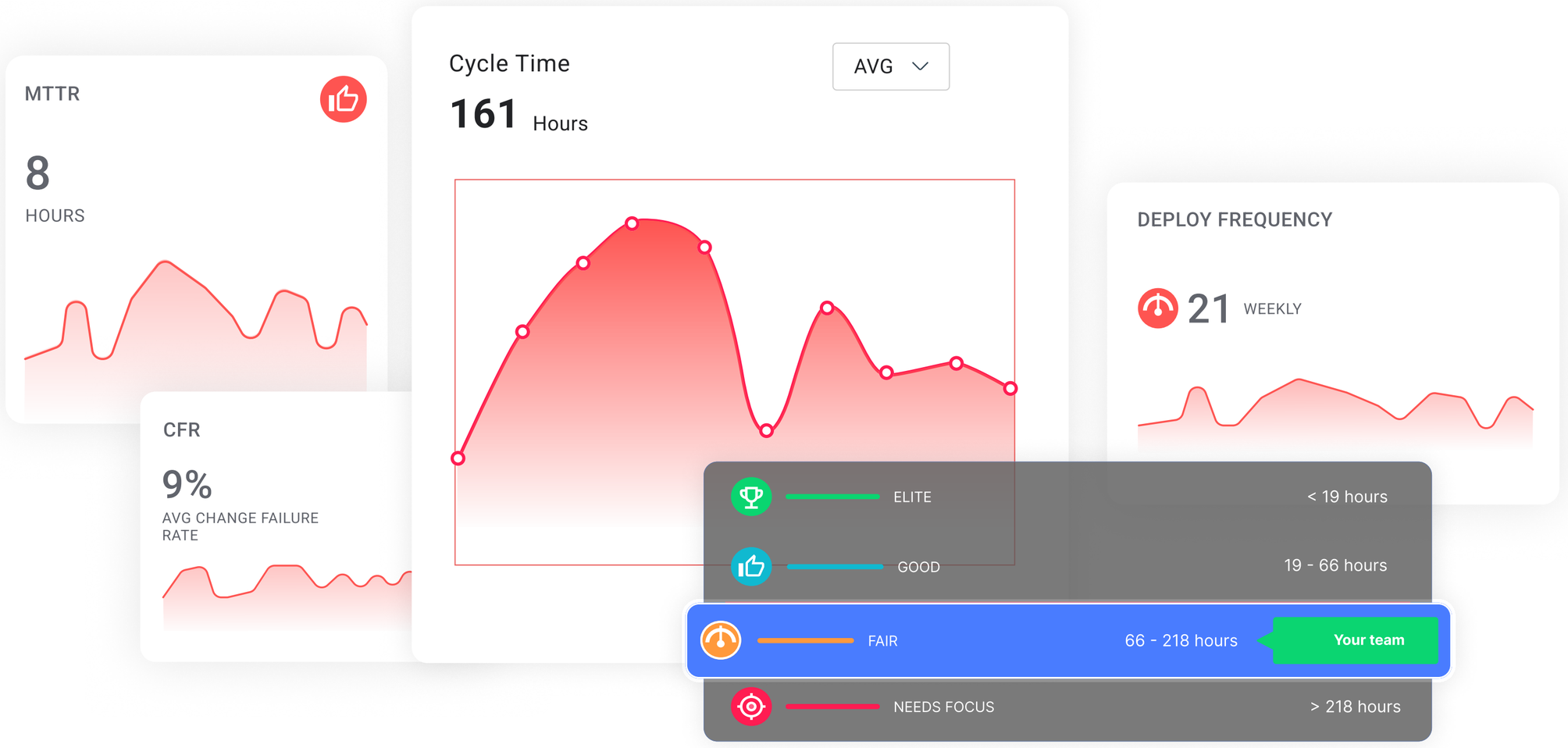
LinearB

MEASURING DEVELOPER PRODUCTIVITY

“But what about DORA and SPACE?”

You’ve likely heard of DORA and SPACE. Both methodologies are valuable and offer key insight into overall productivity.

With DORA, productivity is boiled down to four key metrics that reveal a team’s overall operating efficiency and output stability. These quantitative metrics are revealed through source code management (SCM) tools and issue trackers. SPACE offers both qualitative and quantitative metrics by capturing developer sentiment, collaboration, and communication, as well as more “conventional” KPIs–e.g. DORA metrics.



While both frameworks offer something, they have one key flaw: they only include lagging indicators, i.e., the metrics are calculated and presented after the code makes it into production.

Teams need to focus on both leading and lagging indicators to improve productivity. Take deployment frequency, for example. This lagging indicator DORA metric offers some insight into a team's overall efficiency and, indeed, developer productivity. But short of telling your team to “deploy more frequently,” how do you improve this metric and overall efficiency and productivity?

MEASURING DEVELOPER PRODUCTIVITY

The **Goals, Signals, Metrics (GSM)** approach developed by Google allows for more flexibility and a wider, more complete view of the work to be done to improve productivity. This flexibility means you can use success indicator metrics from other frameworks, like DORA. Using this approach, teams can zoom out and figure out a problem rather than just optimizing for a single metric, get to the root cause of the issue or leading indicator, and work to solve it.

EXAMPLE

GOAL

We want to solve a customer churn problem.

SIGNAL

We deliver customer value (features) more frequently and reliably. Customers resonate with use cases and solutions, log in to our platform more often, and reduce chances of churn.

METRIC(S)

- Reduce PR size to <200 diffs (smaller PRs move through the system faster)
- Shorten cycles (PR pickup, review, approve time)
- Increase merge frequency by 40%
- Increase deployment frequency to production by 30%

The answer lies with leading indicators–in this case, PR size and Merge Frequency. When a team optimizes these metrics, the other metrics--DORA--naturally fall into place. This works for both quality and efficiency measures:

Efficiency: Smaller PRs = Picked up faster = reviewed faster = faster Cycle Time = higher Deployment Frequency

Quality: Smaller PRs = less code to be reviewed = less chance of bugs = lower CFR = lower MTTR

LinearB offers a free forever account that includes the four DORA metrics and the above leading indicators. Get started right here!

MEASURING DEVELOPER PRODUCTIVITY

Four Key Considerations for Developer Productivity

Understanding developer productivity is crucial for distinguishing between mere activity and meaningful progress. This distinction helps ensure that efforts are focused on delivering real value. Productivity in software development goes beyond the simple measure of output. It encompasses the quality, stability, and impact of the work done. True productivity means creating valuable software efficiently and effectively while aligning with business goals and customer needs.

Below you'll find four key considerations that will help you better understand productivity and provide you with ideas to kick-start your productivity improvement initiative.

"True productivity means creating valuable software efficiently and effectively while aligning with business goals and customer needs. ..."

Activity ≠ Productivity

- Lines of Code vs. Meaningful Progress: Writing a large amount of code does not necessarily equate to productivity. True productivity is creating code that solves problems, adds value, and integrates well into the existing system. It's the difference between quantity and quality.

- Busy Work vs. Valuable Work: Engaging in tasks that keep a developer busy but do not contribute to the project's objectives or improve the system's functionality is unproductive. Tasks should be purposeful and aligned with broader goals. For more on this, check out the Engineering Leader's Guide to Goals and Reporting.

MEASURING DEVELOPER PRODUCTIVITY

Alignment to Business Goals

- Misaligned Resourcing: Productivity is not about working hard on tasks that do not align with the company's strategic objectives. Efforts should be directed towards projects and tasks that support and advance the business goals. Read the Guide to Resource Allocation for more on this subject.

- Strategic Contribution: A productive developer understands how their work fits into the larger picture and focuses on tasks that drive the business forward. Make sure developers know how their work will impact the business with frequent check-ins using Investment Strategy data.

Quality and Stability of Work

- Rushed Outputs: Quickly producing work that is low quality or unstable is counterproductive. High-quality, stable code that requires less rework and maintenance is a hallmark of true productivity.

- Technical Debt: Accumulating technical debt by cutting corners might yield short-term gains, but it leads to long-term productivity losses. Sustainable productivity involves balancing speed with maintaining a healthy, manageable codebase. Remember that every line of code and commit is a business decision.

“Quality is never an accident. It is always the result of intelligent effort.”

JOHN RUSKIN

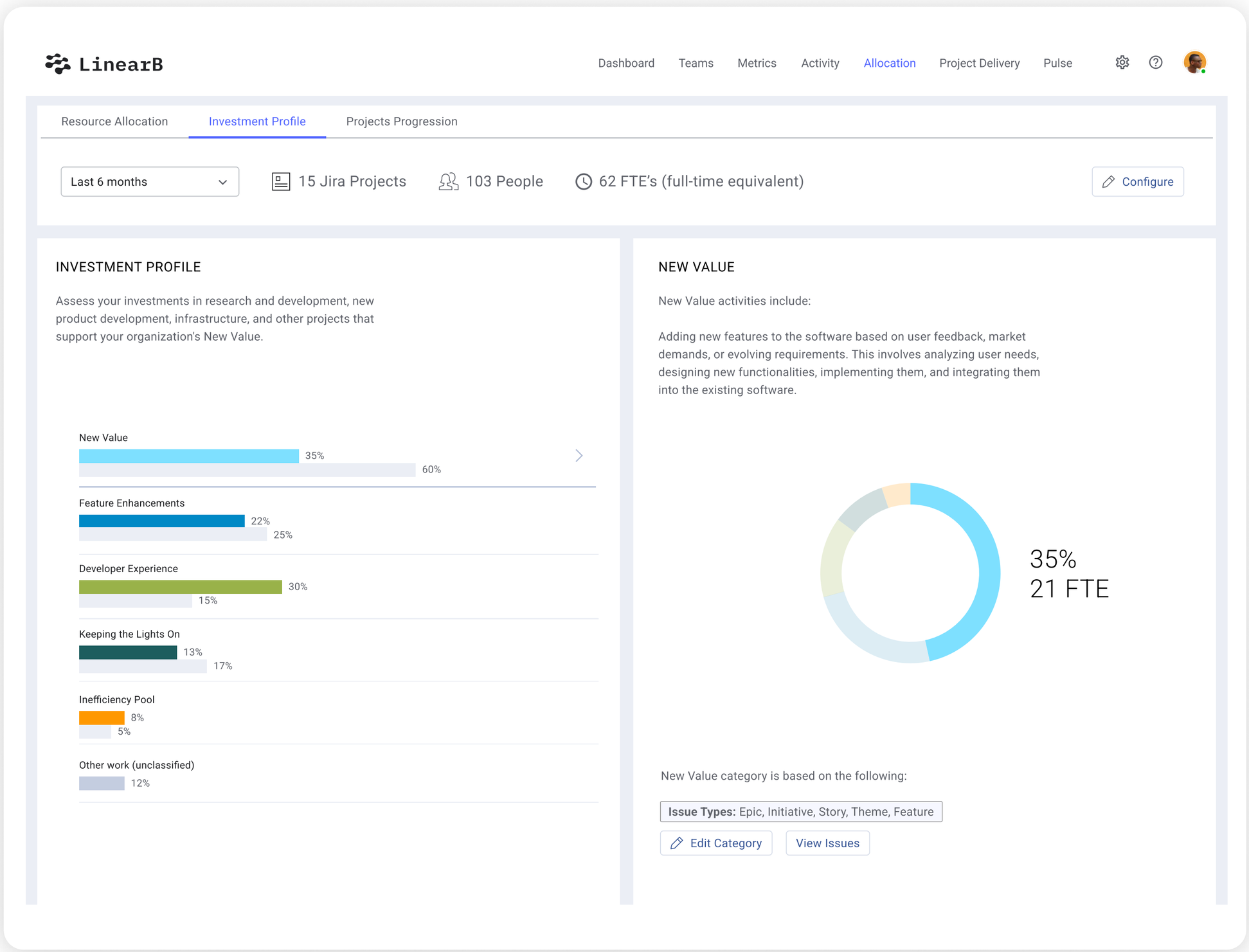
MEASURING DEVELOPER PRODUCTIVITY

Solving Customer Problems and Providing Value

- Customer Needs: Productivity is not just about developing features or fixing bugs but ensuring that the work addresses real customer problems and provides tangible value.

- Enhancing Developer Experience (DevEx): Improving the internal developer experience (tools, processes, environment) is as important as external customer-facing improvements. Productivity includes efforts to enhance DevEx, which in turn accelerates output and improves quality.

- Value Creation: True productivity involves creating new value, whether through new features, enhancements, or maintaining and improving existing systems. It’s about the impact of the work, not just the volume.



LinearB

Accelerating Developer Productivity

Enhancing developer productivity is crucial for a business to maintain a competitive advantage. To achieve this, a structured approach is essential. This section provides step-by-step instructions on accelerating developer productivity by focusing on three key areas:

- Goal setting using quantifiable engineering metrics
- Automating productivity improvement and toil reduction
- Using data to inform decisions, conversations, recurring syncs, and developer ceremonies.

By optimizing for each of these, organizations can create an environment where developers thrive and produce their best work.

Setting Data-Backed Team Goals

By aligning operational goals with business-level OKRs and using metrics to guide the goal-setting process, organizations can ensure that their engineering efforts are both effective and aligned with broader business objectives. This approach not only drives continuous improvement but also reinforces the relevance and impact of the development team's work on the overall success of the organization.

START WITH ORGANIZATIONAL LEVEL OKRS

Establish high-level objectives that align with business priorities. For example, "Increase engineering efficiency by 20% by the end of Q2."

CASCADE GOALS TO TEAMS

Break down organizational OKRs into team-specific goals. Ensure that each team's objectives contribute to the overall organizational goals. For example, "Team A to reduce average PR review time by 25% in the next quarter."

ACCELERATING DEVELOPER PRODUCTIVITY

USE THE SMART FRAMEWORK:

Make sure each goal is Specific, Measurable, Achievable, Relevant, and Time-bound. For example, "Reduce average cycle time from 10 days to 7 days by the end of Q1."

MONITOR PROGRESS AND ADJUST

Regularly track progress against these goals using dashboards and reports. Adjust strategies as needed based on real-time data and feedback.

Aligning Team Goals at Every Level

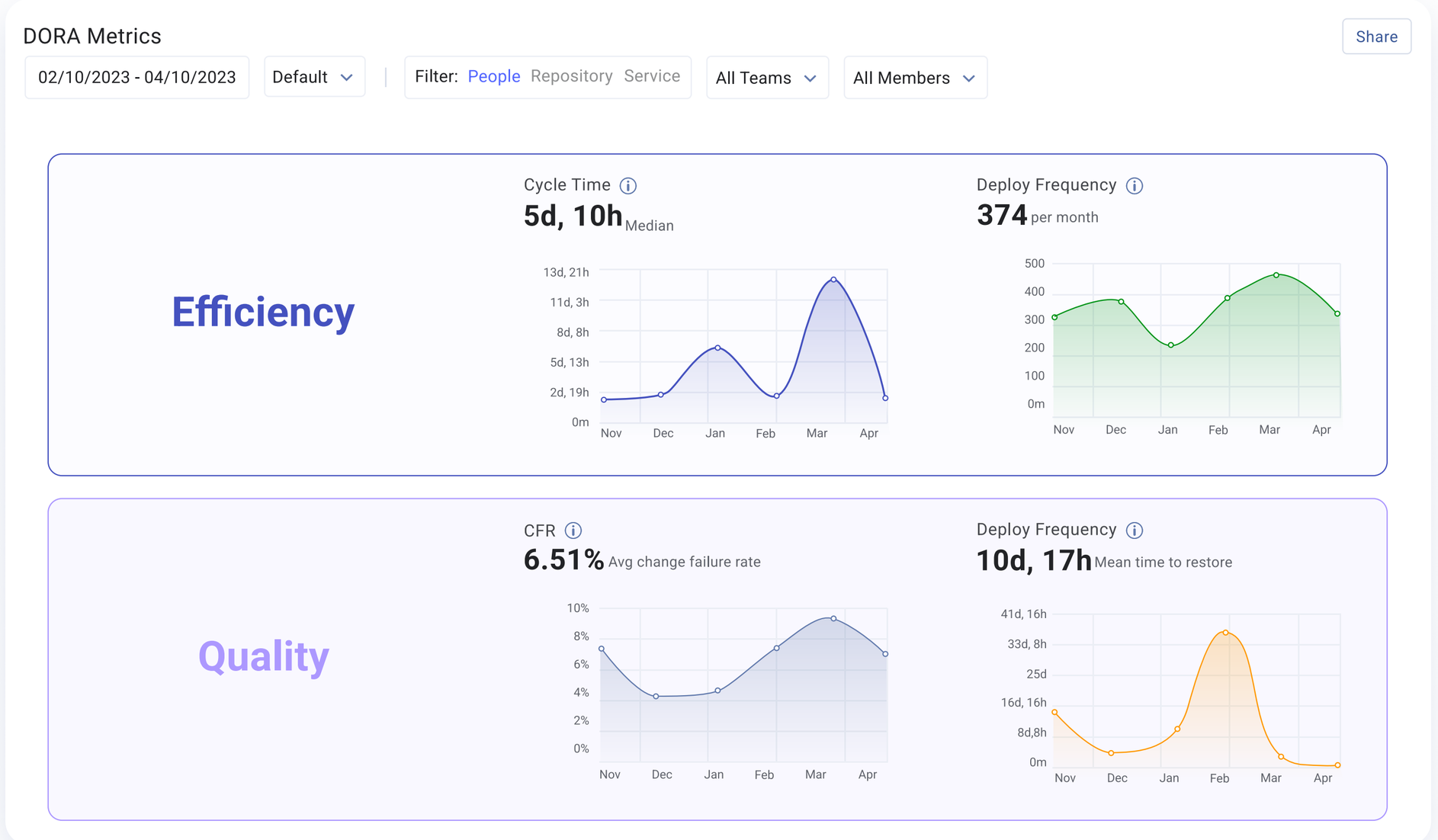
Effective goal setting is foundational for improving engineering execution and aligning R&D with business priorities–both essential components of overall developer productivity. Goals should be anchored around industry-standard measures of engineering performance, such as:

- DORA metrics (Deployment Frequency, Lead Time for Changes, Change Failure Rate, and Time to Restore Service) and leading indicators
- Business alignment KPIs (Resource Allocation and Investment Strategy)
- Project planning, execution, and delivery metrics (like Accuracy Scores and Unplanned/Added Work)
- Engineering benchmarks

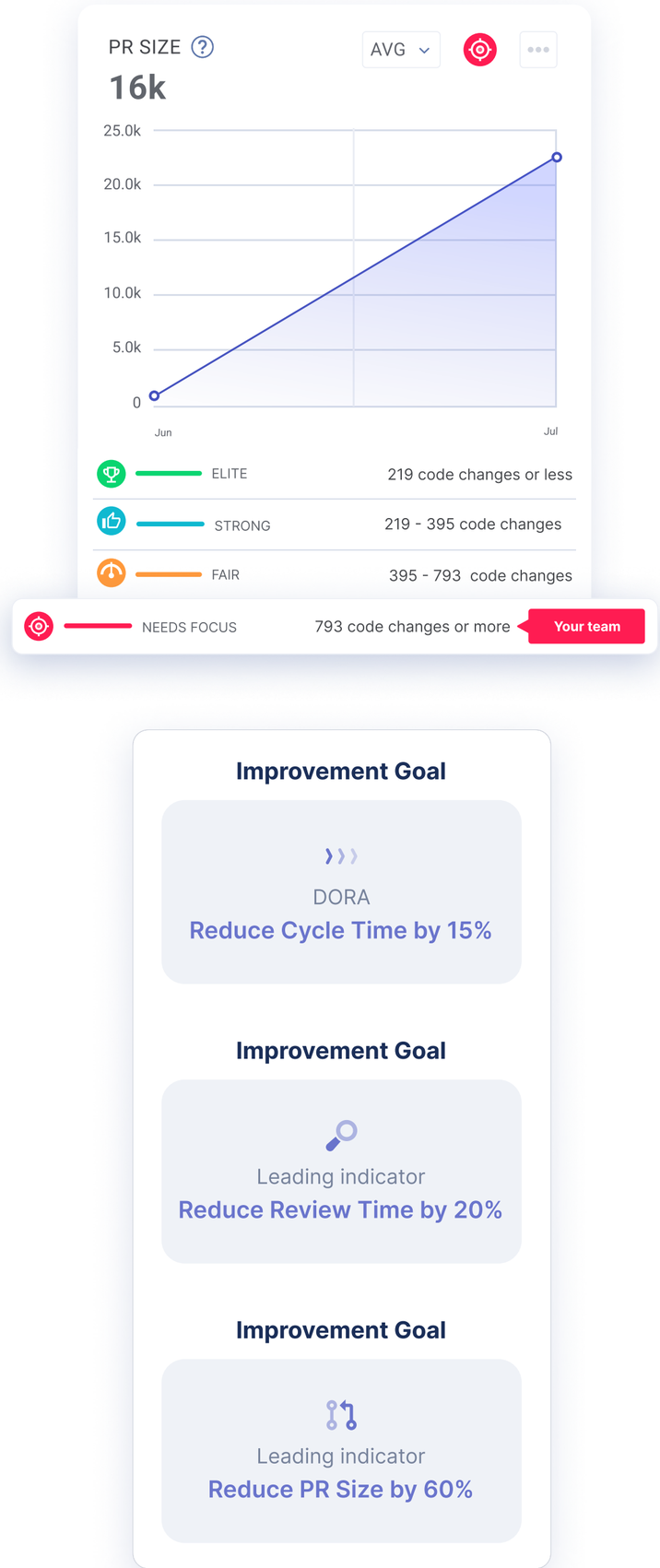
Category	Metric	Elite	Good	Fair	Needs Improvement
Efficiency	Coding Time (hours)	< 0.5	0.5 - 2.5	2.5 - 24	> 24
	PR Pickup Time (hours)	< 1	1 - 3	3 - 14	> 14
	PR Review Time (hours)	< 0.5	0.5 - 3	3 - 18	> 18
	Merge Frequency (per dev/week)	> 2	2 - 1.5	1.5 - 1	< 1
	Deploy Time (hours)	< 3	3 - 69	69 - 197	> 197
DORA	Cycle Time (hours)	< 19	19-66	66 - 218	> 218
	Deploy Frequency	> 0.2	0.2 - .09	.09 - .03	< .03
	Change Failure Rate	< 1%	1% - 8%	8% - 39%	> 39%
	MTTR (hours)	< 7	7 - 9	9 -10	> 10
Quality and Predictability	PR Size (code changes)	> 98	98 - 148	148 - 218	> 218
	Rework Rate	< 2	2% - 5%	5% - 7%	> 7%
	Refactor Rate	< 9%	9% - 15%	15% - 21%	> 21%
	Planning Accuracy	> 85%	85% - 60%	60% - 40%	< 40%
	Capacity Accuracy	Ideal Range 85% - 115%	Under Commit above 130%	Potential Under Commit 116% - 130%	Potential Over Commit 70% - 84%

ACCELERATING DEVELOPER PRODUCTIVITY

Aligning Team Goals at Every Level continued...



To ensure that engineering efforts contribute directly to the overall business success, it's crucial to align operational goals with business-level Objectives and Key Results (OKRs). Start by identifying the core business objectives, such as improving customer satisfaction, increasing revenue, or reducing operational costs. Then, translate these objectives into specific engineering goals that support these outcomes.



For example, if a business objective is to improve customer satisfaction, an operational goal might be to reduce the number of customer-reported bugs by 30%. By doing so, each engineering effort is directly tied to a broader business goal, ensuring that the team's work is relevant and impactful.

ACCELERATING DEVELOPER PRODUCTIVITY

Using Metrics to Find Bottlenecks and Improvement Opportunities

Before setting improvement goals, it's essential to have visibility into current performance and identify areas that need enhancement. Having visibility into metrics, benchmarks, and bottlenecks plays a critical role in this process. Here are some common operational KPIs that teams set goals against:

- **Cycle Time:** Measure the time it takes from starting work on a task to its completion. Long cycle times can indicate inefficiencies in the development process.
- **PR Size:** Track the average size of pull requests. Large PRs might slow down the review process and increase the risk of introducing bugs.
- **Review Time:** Monitor the time it takes to review and merge pull requests. Extended review times indicate larger issues in the development workflow or ways of working.
- **Deployment Frequency:** Assess how often code is deployed to production. Higher frequencies can indicate a more agile and responsive development process.
- **Change Failure Rate:** Evaluate the percentage of changes that result in failures in production. A high change failure rate can highlight issues with code quality or testing processes.

By analyzing these metrics, teams can pinpoint bottlenecks and areas for improvement. For instance, if the review time is consistently high, it may be necessary to streamline the code review process, adopt workflow automation, or allocate more resources to reviews. If the cycle time is lengthy, investigate whether there are delays in specific stages of the development process, such as testing or deployment.

For a deeper dive into goal setting, check out the Engineering Leader's Guide to Goals and Reporting.

Efficiency Metrics

Cycle Time *	4d 1h
Deploy Frequency	9.2/ week
Merge Frequency	19.2/ week
PR Size	359
Coding Time	1h 36m
PR Pickup Time	2d 5h
PR Review Time	1d 19h
Deploy Time	23h

Quality Metrics

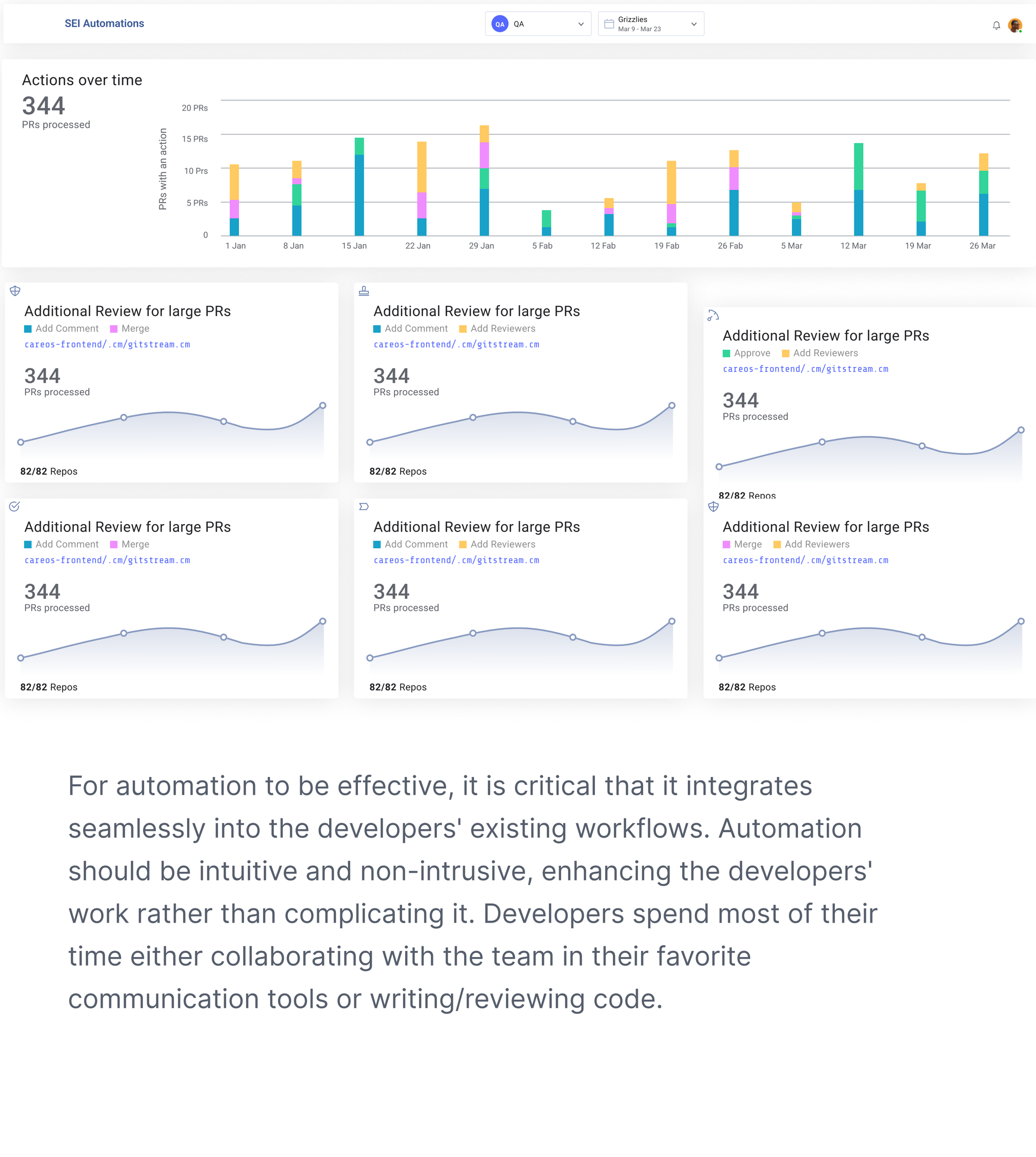
Change Failure Rate*	7.47%
PRs Merged w/o Review	3.7/ week
Review Depth	1.03 comments/PR
PR Size	359
Rework Rate	17.7%
Non-functional work (%)	3.8%
Bug Investment	4%

ACCELERATING DEVELOPER PRODUCTIVITY

Automating Productivity Improvement

Automation plays a pivotal role in accelerating productivity. By streamlining repetitive tasks and reducing manual overhead, developers can focus more on high-value activities that drive business outcomes. This section provides an overview of leveraging programmable workflows and bot assistance to enhance developer productivity. For more on this, check out the Engineering Leader's Guide to Programmable Workflows,

Tools for Toil Reduction and Efficiency Gains

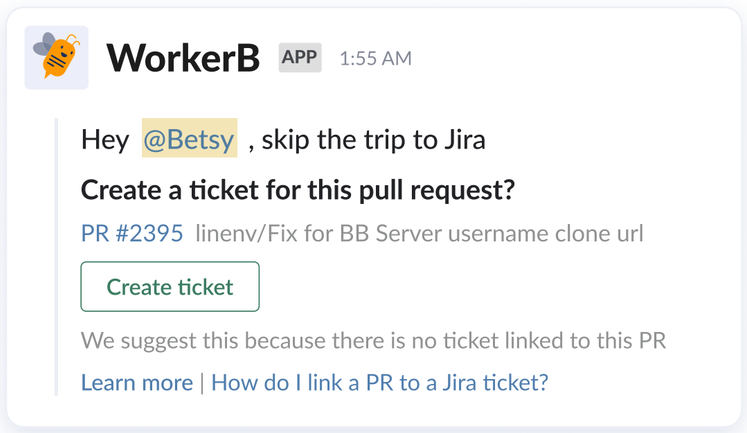


For automation to be effective, it is critical that it integrates seamlessly into the developers' existing workflows. Automation should be intuitive and non-intrusive, enhancing the developers' work rather than complicating it. Developers spend most of their time either collaborating with the team in their favorite communication tools or writing/reviewing code.

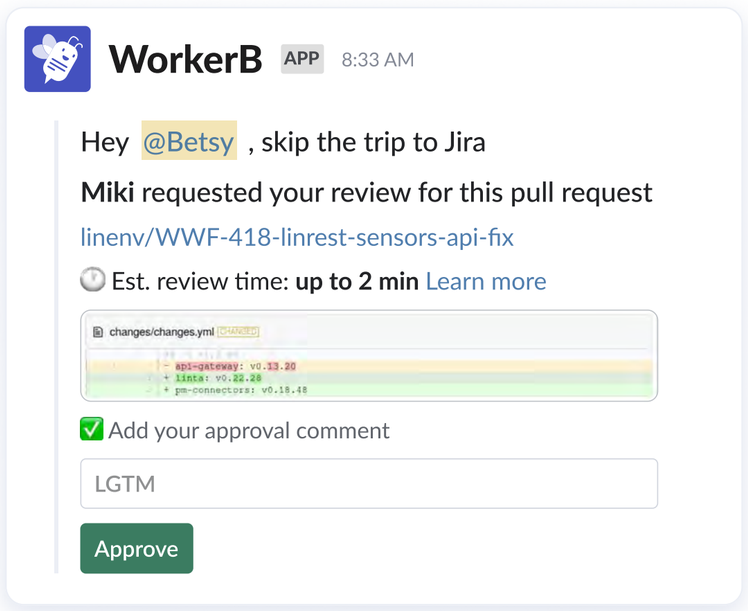
ACCELERATING DEVELOPER PRODUCTIVITY

The following tools meet developers where they are and help with many common tasks:

- **Bot Assistants:** A developer chat bot that automates tedious data entry tasks, speeds up code reviews and approvals, reduces toil/idle time, keeps team focused on any set operational goals, and helps developers prioritize daily tasks.

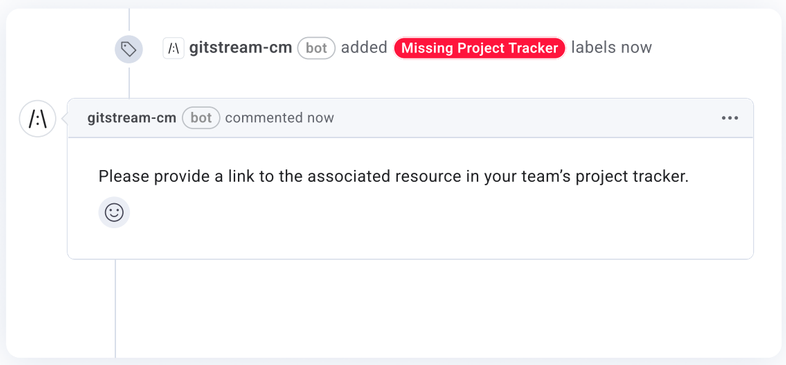


30% of PRs are missing a Jira ticket: automate the process with one-click workflows from Slack or MS Teams

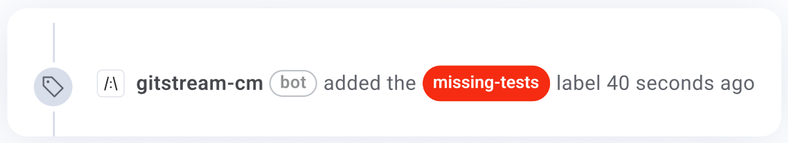


Avoid context switching and review/ approve small PRs directly from Slack

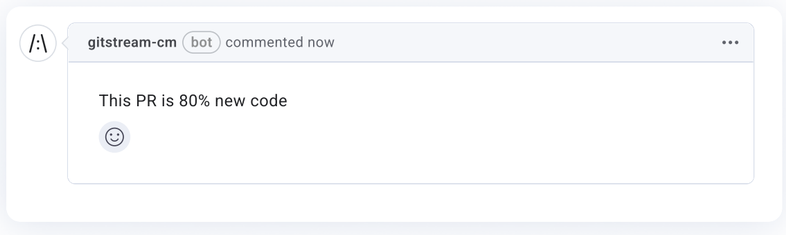
- **Pull Request Orchestration:** Holistic code-level automation and rules engine that can be applied to individual repositories or the entire organization. It classifies and routes pull requests (PRs), assigns the correct reviewer, selectively triggers CI pipelines, automates security checks, adds context for reviewers, automatically approves small PRs and safe changes, and much more.



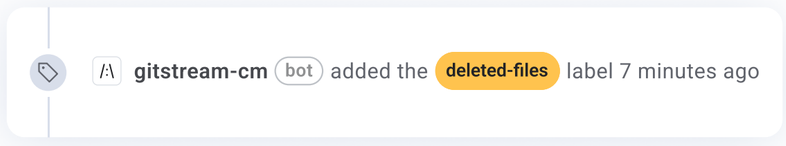
Missing a Jira ticket



Missing testing



This PR is 80% new code



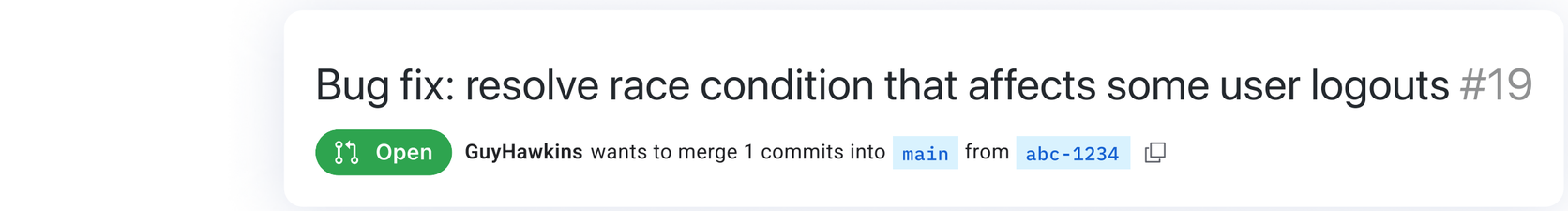
This PR deleted files

ACCELERATING DEVELOPER PRODUCTIVITY

Examples of Automated Workflows

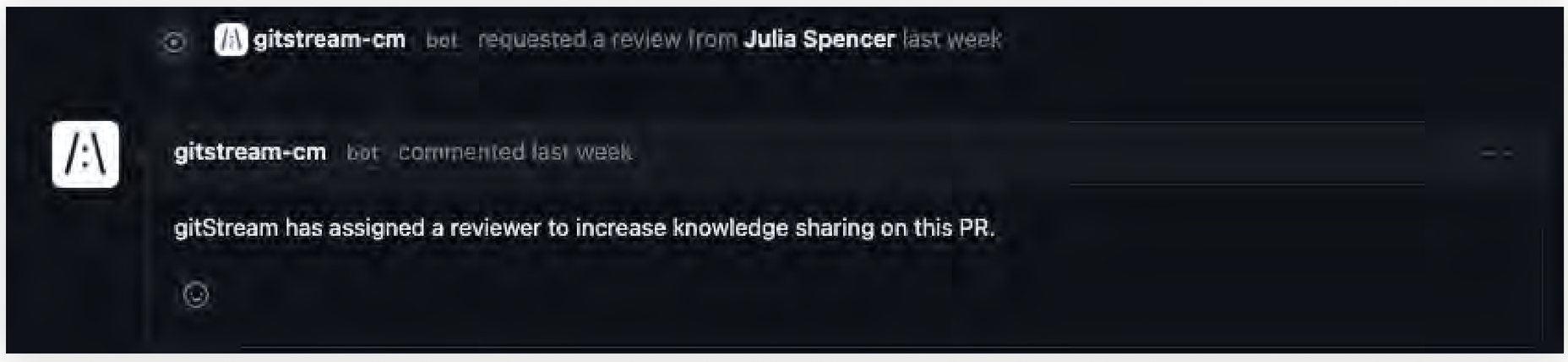
Two ways to drive productivity and find success with automation are to focus on improving efficiency and quality. Here are some examples of how automation can be applied in a development environment to achieve those outcomes:

- **Faster Code Reviews:** Use programmable workflows like CODE EXPERTS to automatically find the best reviewer and assign code reviews accordingly, based on who's most knowledgeable and recently active. This helps improve review quality while decreasing developer toil. To drive efficiency, set up rules to approve small, safe changes like documentation or README.MD files that may not need a full review from a senior developer.

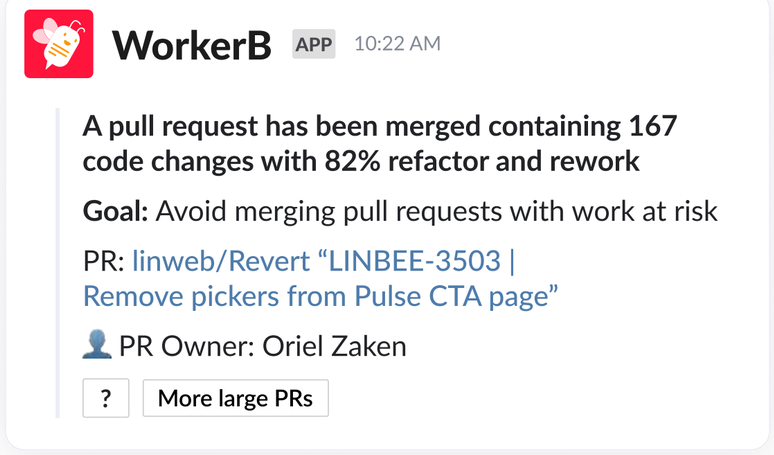


	Wade	Esther
MAR	25 additions, 14 deletions	10 additions, 2 deletions
FEB	127 additions, 46 deletions	
JAN	89 additions, 52 deletions	
DEC	38 additions, 0 deletions	135 additions, 62 deletions
NOV	32 additions, 12 deletions	77 additions, 34 deletions
OCT	15 additions, 2 deletions	45 additions, 52 deletions

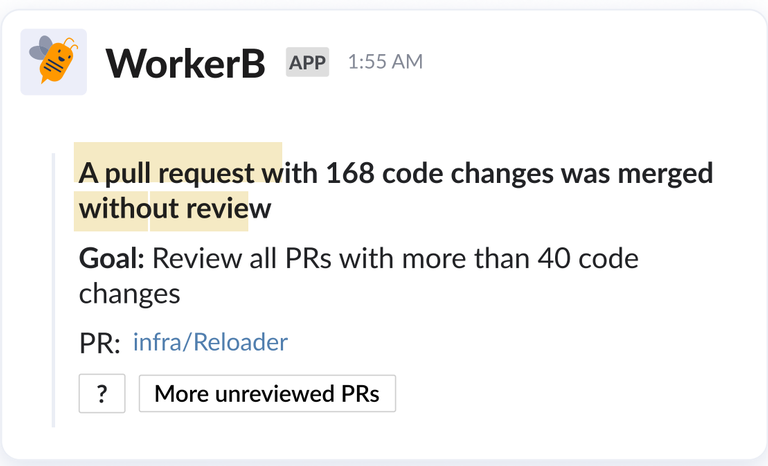
Examples of Automated Workflows



- **Knowledge sharing:** Following the automatic assignment of reviews theme, CODEEXPERTS can also be used here to find the most knowledgeable people who can shed light on particularly complex parts of the code base. In addition to CODEEXPERTS, teams can use the Knowledge Share label to require the reviewer be a previous contributor with code expertise if a PR falls between defined thresholds.



Avoid merging risky work (large PRs with high rework and refactor rates)



Avoid merging PRs that haven’t been reviewed

- **Goal Alignment:** If you’ve identified PR size or Review Time as a bottleneck for your team, you can set automated alerts that fire when a PR is issued that exceeds a certain size or takes too long to review.
- **Security Checks:** Configure SEI+ Automation to perform automated security checks on sensitive PRs, ensuring that the code adheres to security standards before it is merged.

ACCELERATING DEVELOPER PRODUCTIVITY

Best Practices for Automation

To maximize the benefits of automation, follow these best practices:

- **Start Small:** Begin with automating simple tasks and gradually expand to more complex workflows. We recommend things like adding PR labels that provide reviewers with more context.
- **Involve the Team:** Engage the development team in identifying automation opportunities and defining rules. Their input is valuable for creating effective workflows.
- **Maintain Visibility:** Keep dashboards and alerts visible to all team members. This transparency ensures everyone is aware of the automated processes and their impact on productivity.
- **Regular Feedback:** Continuously collect feedback from developers to refine and improve automation rules and workflows for your organization, ensuring they remain aligned with the team's needs and preferences.

Organizations can significantly enhance developer productivity by implementing programmable workflows and leveraging bot assistance.

Automation not only speeds up the development process but also ensures consistency and reduces the cognitive load on developers–allowing them to focus on high-value tasks and drive better business outcomes.

ACCELERATING DEVELOPER PRODUCTIVITY

Decision-Making, Reviews, Syncs, and Ceremonies

Data-driven decision-making is critical for fostering a culture of continuous improvement, providing the team with a better developer experience, and aligning your operational cadences with business goals.

By leveraging data, KPIs, and metrics, managers can establish better habits with their team that lead to more informed conversations, more effective performance reviews, and more productive recurring syncs and developer ceremonies.

For example, prior to reviews, whether quarterly or annually, use the Developer Coaching Dashboard to get data on developers' experience, well-being, and performance.

Informing Conversations and Enhancing Recurring Syncs and Ceremonies

Managers can use productivity data and metrics for more meaningful and actionable team conversations and getting more out of recurring syncs–like daily stand-ups, weekly check-ins, and sprint retrospectives. Here’s how:

- **Setting Context:** Use metrics to provide context during one-on-one meetings with developers. For example, discuss specific KPIs such as PR review time. Be sure to look at this metric in terms of conducting and waiting for reviews or PR size to highlight areas of strength and opportunities for improvement.
- **Identifying Bottlenecks:** Leverage data to identify and address bottlenecks in the development process. For instance, if review times are consistently high, discuss strategies to streamline the code review process with automation. If cycle time is increasing or the project seems to be going off the rails, discuss potential causes, examine risk indicators, and collaboratively develop solutions. These historical metrics can also inform ceremonies like sprint planning or retrospectives. For example, consider past Accuracy scores and team cycle time to set realistic sprint goals and capacity estimates.

ACCELERATING DEVELOPER PRODUCTIVITY

- **Goal Alignment:** Ensure individual goals are aligned with team and organizational objectives. Use data to show how individual contributions impact broader goals, fostering a sense of ownership and accountability.
- **Celebrating Success:** Use data to recognize and celebrate team achievements. Highlight metrics that show improvement and acknowledge contributions that led to success.

Conducting Performance Reviews

Performance reviews should be based on objective data to ensure fairness and transparency however, they should also incorporate context and sentiment. Here's how to integrate metrics into performance reviews:

- **Objective Metrics:** To assess performance, use quantifiable metrics such as deployment frequency, code quality (e.g., change failure rate, rework), and contribution to team goals.
- **Balanced Evaluation:** To provide a balanced evaluation of performance, consider a mix of efficiency and quality metrics and remember to account for seniority as well as the different roles a developer plays on a given day. For example, if you're having a conversation with a more senior developer, make sure to bring data on review time and reviews conducted, in addition to merges and PR size, as well as what areas of the code base they've touched, workload data, and more. You'll then have a more complete picture of the overall productivity of that developer.
- **Continuous Feedback:** Provide ongoing metrics-based feedback, not just during the formal review. This helps developers continuously improve and align their efforts with expectations.

LinearB

ACCELERATING DEVELOPER PRODUCTIVITY

Enhancing Regular Conversations and Check-Ins with Data

While data and metrics provide valuable insights, they should enhance regular conversations and check-ins with developers, not replace them. Here's how to integrate data into these interactions effectively:

- Complement, Don't Replace: Use metrics to augment the qualitative aspects of your conversations. Metrics provide a factual basis for discussion, but personal interactions give context and understanding that data alone cannot provide.
- Foster Open Communication: Encourage open dialogue where developers can share their perspectives on the data. This can lead to a better understanding of the underlying causes of certain metrics and foster collaborative problem-solving.
- Provide Personalized Feedback: Use data to provide specific and actionable personalized feedback. Discuss individual contributions and areas for improvement in the context of the team's overall performance.
- Emphasize Support and Development: Use metrics to identify areas where developers might need additional support or training. This can help in tailoring development plans and providing resources that align with their growth needs. It is also a sure-fire way to improve the developer experience.
- Build Trust: Be transparent about how data is used. Ensure developers understand that metrics are tools for improvement and not for monitoring individual performance. This builds trust and encourages a more positive attitude towards data-driven practices.

This approach not only enhances developer productivity and improves DevEx, but also ensures that teams are focused on delivering high-quality software that drives business success.

ACCELERATING DEVELOPER PRODUCTIVITY

Reporting On Developer Productivity

Effective reporting on developer productivity is crucial for communicating progress, identifying areas for improvement, and aligning efforts with business objectives. Audience-specific reporting tailors the presentation of data and insights to meet the needs of different stakeholders, ensuring that each group receives the most relevant and actionable information. Here's how to create reports for developer productivity that will resonate with various organizational stakeholders.

Senior Leadership and the C-Suite

Senior leadership and C-suite executives are primarily concerned with how developer productivity impacts the business–like delivering customer value more quickly, driving costs down via efficiency gains, or getting more high priority projects completed. Reports for this audience should focus on high-level metrics and strategic outcomes related to productivity:

- **Key Deliverables and Milestones:** Provide updates on the status of key projects and milestones, highlighting how improvements in developer productivity have accelerated timelines, improved planning/capacity accuracy scores, or increased efficiency.
- **Business Impact:** Connect productivity metrics to business outcomes. For example, demonstrate how reducing cycle time has improved time-to-market for new features or how increasing deployment frequency has enhanced product reliability and customer satisfaction.

Senior Leadership and the C-Suite continued...

- **Resource Allocation:** Show how improved productivity has optimized resource allocation across projects and/or how it has helped optimize engineering's investment mix. For example, if your team was more productive and efficient and you were able to reallocate headcount/cycles to high-priority projects, start a new project ahead of schedule, or invest more in work categorized as new value that is likely to drive revenue, CSAT, and retention.
- **Risk Management:** Identify any risks or challenges that could impact business objectives due to productivity issues. Provide a mitigation plan and highlight areas where additional support or resources are needed.

Be sure to outline strategic initiatives and potential investments in tools or training to sustain or enhance productivity. Highlight the importance of aligning productivity improvements with business goals and resource allocation to support these initiatives.

Key Engineering Investments

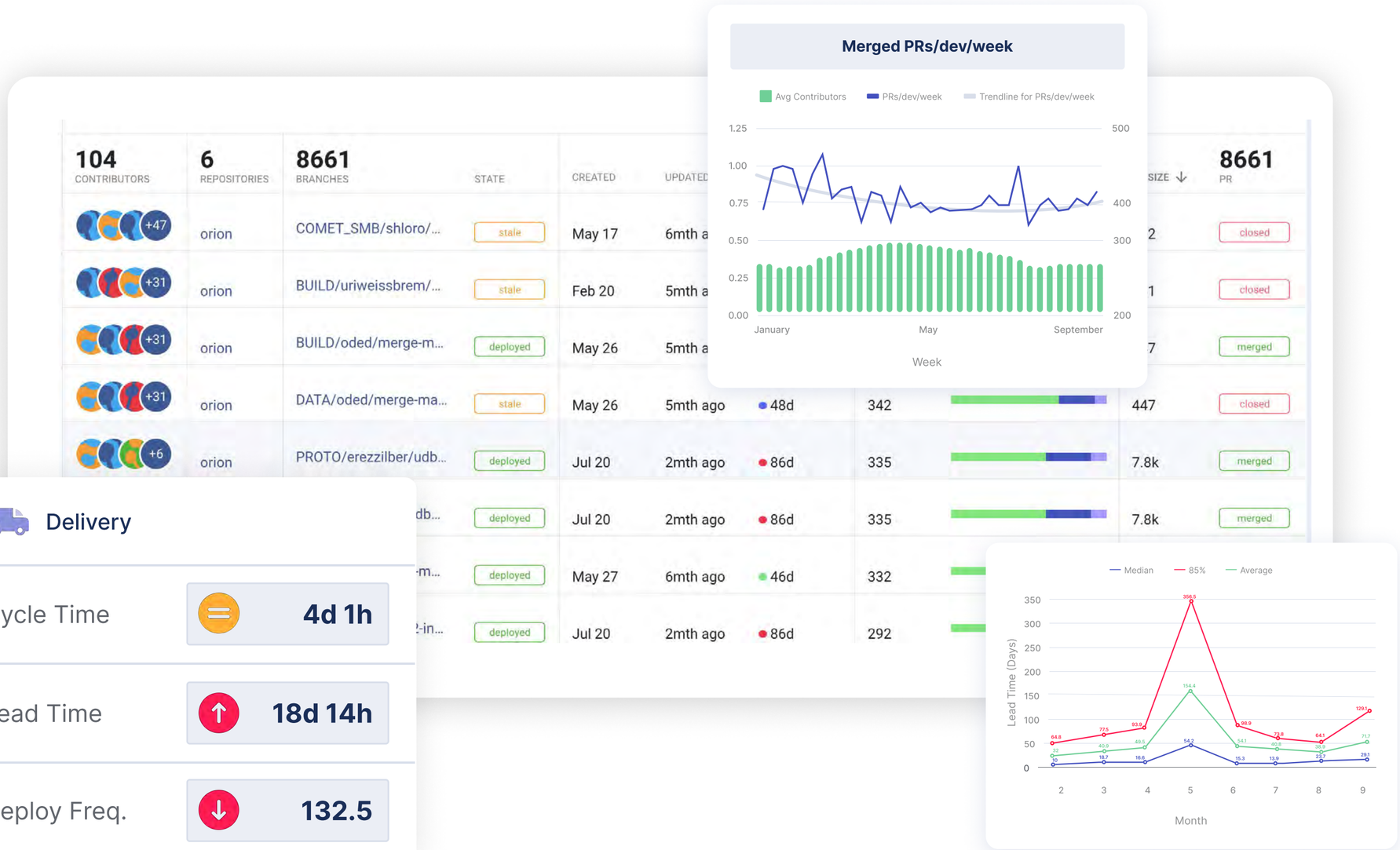
Q1, 2023

Initiative	Business Impact	Q1 Actual		Engineering Execution			Q2 Recommend
ABC-123 Lorem ipsum dolor sit amet	HIGH	8.4	**$315**K	STRONG	4d 1h	1.8	Boost
PRJ-14 Morbi id purus vitae risus sagittis	HIGH	7.8	**$293**K	FAIR	6d 2h	1.1	Improve
MOB-456 Morbi id purus vitae risus sagittis	MED	7.5	**$281**K	ELITE	1d 7h	2.9	Maintain
ABC-223 Integer gravida erat vitae rutrum	LOW	6.2	**$233**K	NEEDS FOCUS	10d 2h	0.8	Reduce
CS-111 Sed feugiat enim elementum	HIGH	–	–	–	–	–	Start
<All other work>		7.6					

FTE | Estimated Cost (avg. cost of $150K/dev) | Cycle Time | Merge Frequency (Merged PRs/Dev/Week)

ACCELERATING DEVELOPER PRODUCTIVITY

Engineering Managers and Team Leads



Engineering managers and team leads need detailed insights into team productivity, skills and competencies, and process efficiency to manage day-to-day operations and drive improvements–this data is the key to providing a superlative DevEx. DevEx underpins efficiency, retention, and morale–all of which inform productivity:

- **Team Productivity Metrics:** Include detailed metrics such as cycle time, PR review time, deployment frequency, and change failure rate. Highlight trends and identify areas for improvement.
- **DevEx Metrics:** Look into trends across teams to get a sense of the developer experience within your organization. Determine if opportunities exist to expose devs to new technologies and skills or break the monotony of only working on one thing. This is also a good time to see if workloads are balanced and you're mitigating risk of burnout. Other things to consider include:
  - **Toil:** Are Pickup, Review, and Wait times higher than average for your org or as compared to the industry?
  - **Dev Metrics:** Is everyone on your team contributing (i.e. authoring and reviewing code) as expected for their role and seniority?
  - **Skills and Competencies:** Are you cross-training and exposing your team to new technologies, languages, and frameworks? Are they adopting and implementing them?

ACCELERATING DEVELOPER PRODUCTIVITY

Engineering Managers and Team Leads

If you've identified gaps here, come up with a plan to address them like setting Team Goals, implementing automation, adopting new tools, or looking into training.

- **Bottlenecks and Process Improvements:** Use data to pinpoint bottlenecks in the development process that affect productivity. Provide actionable insights and recommendations for process improvements.
- **Goal Progress:** Track progress against team-specific productivity goals and OKRs. Show how individual and team efforts contribute to broader organizational productivity objectives.
- **Workload and Capacity:** Provide visibility into team workload and capacity. Highlight any imbalances and suggest adjustments to ensure sustainable productivity and prevent burnout.

Be sure to suggest practical steps for addressing identified bottlenecks and improving processes, such as changes to workflows and adjusting ways of working. Emphasize the need for continuous monitoring and adjustment based on data-driven insights to maintain progress toward productivity goals.

Individual Developers

Individual developers benefit from personalized feedback that helps them understand their contributions to team productivity and areas for growth. It's also a chance for you to enhance DevEx data with supporting anecdotal evidence to get a more complete picture.

ACCELERATING DEVELOPER PRODUCTIVITY

Individual Developers continued...

Remember that this reporting should be a two-way street and more of a conversation that accounts for nuance rather than a rigid readout of metrics:

- **Personal Productivity Metrics:** Provide metrics that reflect individual contributions to productivity, such as PR size, review time, and number of deployments. Use these metrics to give specific, actionable feedback.
- **DevEx Metrics:** Provide data that paints a picture of their collaboration habits, like authoring and reviewing code. See if toil is a significant blocker for them. Ask, “are they waiting a lot longer than the rest of the team for a review?”. Look into whether they have a balanced workload. See if they’re working on projects that align with skills and competencies. Look into WIP and ensure it’s at healthy levels.

ACCELERATING DEVELOPER PRODUCTIVITY

Individual Developers continued...

- Development Goals: Align individual productivity metrics with personal development goals and career growth plans. Highlight areas of strength and opportunities for skill development.
- Recognition and Support: Recognize individual achievements contributing to overall productivity and provide support where needed. Use data to identify areas where additional training or resources might be beneficial.
- Continuous Improvement: Encourage a culture of continuous improvement by regularly discussing productivity metrics and setting short-term goals for individual growth and development.

Use this data to provide personalized next steps for improving productivity, including targeted training, mentorship opportunities, or specific tasks designed to enhance skills and efficiency. Focus on establishing a supportive environment that encourages personal and professional growth, aligning individual development goals with overall team productivity objectives.

CONCLUSION

LinearB Helps You Improve Developer Productivity

LinearB is a powerful platform designed to enhance developer productivity by providing robust capabilities for measuring and benchmarking teams using operational, project delivery, as well as resourcing and investment strategy metrics. LinearB makes it easy to identify bottlenecks and set improvement goals based on historical trends, project forecasting insights, and industry data.

It also ensures the success of improvement initiatives by providing automation workflows at every level to help developers reduce toil, maximize flow state, offload tedious manual tasks, prioritize their time, enjoy a great DevEx, and get back to what they love–being productive and solving problems with code.

Ready to enhance your team's productivity? Sign up for a free LinearB account or book a demo to see how the platform can transform your development process and dial developer productivity up to 11.

LinearB

CONCLUSION

Congratulations, you're now well on your way to optimizing every aspect of developer productivity. To recap, we covered:

- **Measuring Developer Productivity:** We discussed the benefits and common challenges associated with measuring developer productivity.
- **Productivity KPIs:** We covered what metrics and KPIs are the best leading/lagging indicators of developer productivity and the functional areas to focus improvement efforts–efficiency, effectiveness, and experience. We also defined productivity in the context of the SDLC.
- **Accelerating Developer Productivity:** We outlined the step-by-step process for improving developer productivity within your organization. As a reminder, the process is as follows:
  - Baselining your teams and setting data-backed improvement goals
  - Using automation to meet those goals
  - Leveraging this telemetry to inform and enhance conversations with your team
  - Establishing a regular cadence of audience-specific reporting, alignment, and recommendations to further improve

LinearB

CONCLUSION

Begin implementing the strategies discussed in this guide to see tangible improvements in your team's productivity. Whether it's setting SMART goals, automating workflows, or leveraging data-driven insights, taking action now will set you on the path to continuously improving developer productivity.

This guide is part of a series dedicated to engineering excellence. Stay tuned for the next installment, where we'll explore advanced strategies for improving the delivery predictability of your engineering organization.

Want to continue learning? Explore additional resources and tools to support your productivity enhancement journey. Check out the following:

- LinearB Blog for insights, feature spotlights, and best practices.
- Engineering Benchmarks to help you define “good” for your team with data and insight.
- Workflow Automation Tools to streamline your processes.

LinearB