educative

IGNITING DEVELOPER PRODUCTIVITY

HOW THE BEST ENGINEERING LEADERS FOSTER EFFECTIVE COLLABORATION

HOW ENGINEERING LEADERS FOSTER HEALTHY AND PRODUCTIVE TEAMS

When engineers transition to lead and manager roles, it can sometimes be difficult to adjust to new priorities and expectations. As a developer, your performance is measured by the quality of your code. As an EM, your performance is now measured by the impact of your teams. Adapting to these metrics means focusing less on technology and more on people. You are now responsible for helping teams with unique skills meet company goals.

This shift in focus means your success as a leader is concerned with the question of: how can you get the most out of every developer to grow healthy and productive teams?

One important aspect of building successful teams is equipping developers with the technical skills to grow your product. Fortunately, there are learning platforms and courses available to help developers keep up with new technology. Integrating these tools into your team's workflow is crucial to your success.

However, strong technical skills and up-to-date knowledge will only get your teams so far. Developers can be well-versed in the latest languages and technologies. But if they don't have the interpersonal skills to work together effectively, team health and productivity will suffer.

The best engineering managers approach developer growth holistically. When we spoke with engineering leaders from across the industry, and they shared the most important qualities of their teams, one non-technical quality came up over and over: collaboration. They understand that technical skills and effective collaboration work hand-in-hand to accelerate your teams. They also understand that collaboration, like any set of technical skills, can be taught. With developers who are willing to grow, you can implement management practices that yield more productive working relationships.

To help you unlock your developers' full potential, let's start by:

1. **Defining the qualities of highly collaborative teams and how you can establish them**

2. **Understanding how to implement management practices that increase developer collaboration**

Of course, your teams won't become more collaborative overnight, and there are no shortcuts to building high-performing teams. That's why it's all the more important to invest in long-term developer growth today.

Here's what we'll cover:

Contents

HOW TO ESTABLISH THE QUALITIES OF HIGHLY COLLABORATIVE TEAMS

We know that the most productive developer teams must have two essential skill sets:

1. **The technical skills to keep your product growing**

2. **The interpersonal skills to collaborate together effectively**

Defining a technically equipped team is straightforward: are your developers skilled enough in the latest language or technology to take your product to the next level? Developer learning platforms make it easy to identify and address these skill gaps (something we are all too familiar with here at Educative).

On the other hand, defining a highly collaborative team is more challenging.

We've talked with world-class engineering leaders to find out what effective collaboration looks like on their developer teams. Based on these conversations, we've identified 3 key qualities of highly collaborative teams:

1. **Trust**

2. **Shared Purpose**

3. **Clear Processes**

5 TIPS TO BUILD A FOUNDATION OF TRUST

1

ASK RESPECTFUL QUESTIONS THAT ADVANCE THE CONVERSATION.

When engaging with your team's ideas, avoid asking "why." Instead, focus on "how" or "what" questions. They will prompt more specific explanations and lead to more productive discussions.

2

DON'T BE AFRAID TO BE VULNERABLE.

Share your mistakes and the lessons learned with your team. This will encourage developers to adopt a growth mindset.

3

DON'T MISTAKE SILENCE FOR AGREEMENT.

Take silence as an opportunity to pose deeper questions and learn more about your team's needs.

4

ACKNOWLEDGE THAT NEW PROJECTS HAVE RISKS.

Expect challenges and treat failure as a learning experience.

5

MAKE TIME FOR TEAM BUILDING.

Getting to know team members on a personal level builds stronger working relationships and ultimately improves your team's output.

> ***Psychological safety is a climate in which you feel safe to be yourself and grow from your mistakes (personal or technical).***
>
> ***– Jossie Haines***
> *Leadership consultant, former VP of Engineering at Tile*

Let's discuss what each quality looks like in practice. Then we will explore how you can establish these qualities on your team.

1. Trust

Imagine an environment where the team only brings "safe" ideas. Nobody speaks up if they realize they've made a mistake (or spotted someone else's). Nobody asks clarifying questions for fear of being labeled as someone who doesn't know their stuff.

This low-trust climate is not conducive to building impactful products and services.

Jossie Haines, leadership consultant and former VP of Engineering at Tile, encourages engineering managers to build trust through **psychological safety.**

Jossie's recommendations are based on extensive data. In a two-year study on team performance, Google found that psychological safety is the single most important quality that distinguishes successful teams.

That's because psychological safety is an evolutionary need. When a team member dismisses your ideas or makes belittling comments, your brain processes this as a threat. The amygdala activates your flight-or-fight response, shutting off access to higher cognitive functions.

When engineers don't feel safe, they can't be productive – and certainly won't stick around for long.

In a psychologically safe environment, developers trust that they will not be punished for their mistakes. This confidence empowers everyone to take intellectual risks and speak candidly – behaviors that are necessary for productive collaboration. When you have a strong foundation of trust, you are more likely to see high productivity and high retention.

Here are some general tips for building trust through psychological safety:

- **When engaging with your team's ideas, ask respectful questions that advance the conversation.** Instead of asking "why," focus more on "how" or "what" questions.
- **Don't be afraid to be vulnerable.** Share your mistakes and the lessons learned.
- **Don't mistake silence for agreement.** Take silence as an opportunity to pose deeper questions.

- **Acknowledge that new projects have uncertainty and risk.** Expect challenges and treat them as learning experiences.

- **Create time for team-building activities, especially in remote environments.** This builds a foundation of trust that yields stronger working relationships – and ultimately improves your output.

> ***Engaging in open dialogue about technical and business trajectories can help everyone to align on product goals and collaborate more effectively.***
>
> **– Clement Kao**
> *PM and founder of ProductTeacher*

2. Shared purpose

Is your team more focused on shipping tickets than solving user pain? If so, you may be working in a feature factory.

Feature factories are teams that generate output without understanding the business impact of their contributions.

This lack of connection to a shared purpose:

1. **Diminishes motivation on your team**

2. **Inhibits collaboration by keeping developers siloed**

Clement Kao, a Product Manager and founder of ProductTeacher, knows that it can be tempting to dive into work without considering its broader impact. Everyone wants to hit their deadlines. However, he encourages EMs to invest time in defining high-level goals, especially when you are working cross-functionally.

Although this dialogue takes more time upfront, it enables better collaboration that will accelerate your team in the long term.

For Magda Miu, an Engineering Manager at Adobe, defining shared purpose is one of the first steps to building motivated and collaborative teams: “As a leader, your role is to connect everyday tasks with the larger question: Why does my team exist?”

When employees understand how their contributions impact the mission of your team and company, they will feel more engaged and inspired.

This commitment to shared business goals helps developers stay focused and push themselves as a team.

Magda helps developers become purpose-driven through working agreements. These are documented behaviors that align with your team’s values and help you achieve your mission at the company. Working agreements help to establish behavioral norms in all areas of your work.

For example, let's say one of your values is ***user focus: meeting/ exceeding customer needs and desires.*** The working agreement outlines how you can embody this value as a team. Here are some examples you can include in your working agreement:

- *Ask for user feedback and include it in the work*
- *Learn from failures and avoid repeating them*
- *Focus on quality, technical excellence, and outstanding customer experience*

From there, define OKRs for your team's working agreement and update them monthly or quarterly. For example:

Working agreement: Learn from failures and avoid repeating them

OKRs:

- **Make 3 improvements to team processes**
- **Document 1 "failure story" and share it with the team**

These should be documented in a knowledge base that everyone can access.

At the end of each month, your team can:

- **Update the status of each item**
- **Discuss whether it is still relevant for your context**
- **Revise accordingly**

Making behavioral expectations clear and embedding them in your team's daily work builds accountability. In a climate where teams uphold their working agreements, everyone will feel more empowered to take intellectual and creative risks.

3. Clear processes

When you hear "team processes," you likely think about coding guidelines or project management workflows. Magda Miu encourages engineering managers to create similar processes around collaboration.

> ***"Leaders have expectations around how people communicate, address conflict, and manage their time. However, we don't always make those expectations clear. Instead of hoping people will figure it out on their own, be transparent about the high-performing mindset you want to instill and the processes that go along with it."***
>
> **– Magda Miu**
> *Engineering Manager at Adobe*

4 STEPS TOWARD CREATING A WORKING AGREEMENT WITH YOUR TEAM

1 DEFINE YOUR TEAM'S WORKING AGREEMENT

Talk with your team to brainstorm behaviors that will help you achieve your mission

2 DEFINE THE ACTIONABLE STEPS (OKR'S) THAT YOU AND YOUR TEAM WILL TAKE TO UPHOLD THIS AGREEMENT.

- Make 3 improvements to team processes.
- Document 1 "failure story" and share it with the team.

3 ENSURE YOUR WRITTEN OKRS ARE VISIBLE AND ACCESSIBLE TO EVERYBODY ON THE TEAM.

-Have your team document their contributions to OKRs over the next month
- Document-sharing space that is easily accessible to all members of the team

4 AT THE END OF EACH MONTH, YOUR TEAM CAN EVALUATE YOUR OKR'S BY:

Updating the status of each item.
Discussing whether it is still relevant for your context.
Revising and editing accordingly.

Clear technical processes help developers to meet team standards for engineering excellence. Similarly, clear interpersonal processes help developers collaborate more effectively.

To get started, focus on creating guidelines for the following:

- How to navigate disagreement
- How to deliver and receive feedback
- How to communicate efficiently

Of course, imposing processes on a team without their buy-in is counterproductive to building a healthy culture. To help everyone be invested and accountable, work with your team to create processes that meet their needs. Make it clear that these processes are revisable: iterate periodically to match the present context.

To ensure that processes are well integrated in your team's workflow, it's important to consolidate them in a central knowledge base that everyone can access.

When documentation is scattered across platforms as README files or links to resources, developers have to spend more time looking for the information they need. They will also be less likely to implement the processes you've created.

BENEFITS OF IMPLEMENTING OKRS

- *Holds leaders and team members accountable*
- *Clear goals encourage team members to take more creative and intellectual risks*
- *Clearly defines goals, so there is no confusion or miscommunication between team members and their managers.*

Review: How to develop clear processes with your team

1. Identify expectations for team communication. For example:
    - Navigate disagreement with respect.
    - Deliver actionable feedback and receive it gracefully.
    - Communicate information efficiently and in the appropriate channels.
2. Work with your team to outline processes that achieve these outcomes.
3. Document these processes in a consolidated knowledge base.
4. Revisit documentation processes periodically and iterate to match the present context.

3 PILLARS

FOR CREATING EFFECTIVE COMMUNICATION GUIDELINES AMONG YOUR TEAM

1. HOW TO NAVIGATE A DISAGREEMENT

Agree on an actionable and documented process for when disagreements emerge.

2. HOW TO DELIVER AND RECEIVE FEEDBACK

Clearly outline strategies and expectations when receiving constructive criticism and how to communicate with others when they might need some helpful feedback.

3. HOW TO COMMUNICATE EFFICIENTLY

Set expectations and guidelines for how you'd like your team to communicate on a regular basis with you and among themselves.

4 STEPS TO DEVELOPING CLEAR PROCESSES WITH YOUR TEAM:

1. Identify clear expectations for team communication
2. Work with your team to outline processes that achieve these communication goals
3. Document these processes in a consolidated knowledge base
4. Revisit documented processes periodically and iterate to match the present context

How to Improve Collaboration for More Productive Teams

Just as developers can build technical knowledge over time, they can learn to become effective collaborators.

Once you have created interpersonal norms for your team through working agreements and processes, it's time to help each developer grow towards these expectations.

We will discuss strategies to support developer growth in two phases:

1. **Onboarding:** lays the groundwork for interpersonal norms on your team

2. **Growth plans:** support developers in leveling up their communication skills to achieve your team's mission

Onboarding

As you build collaboration on your team, you'll also need a plan for integrating new hires into your culture.

That's because increasing the size of your team doesn't automatically accelerate productivity.

You need processes and practices that enable success – and methods of communicating them at scale.

INCREASING TEAM SIZE

+

EFFECTIVE ONBOARDING PROCESSES AND COMMUNICATION

=

ACCELERATED PRODUCTIVITY.

According to the 2022 Stack Overflow developer survey, over 48% of all respondents think onboarding takes a very long time or a somewhat long time at their organization. Effective onboarding helps new hires ramp up, reach productivity quickly, and stick around longer.

We'll cover four steps to onboard new developers efficiently and maximize team performance with every new hire:

1. Build a bridge between hiring and onboarding.
2. Overcommunicate.
3. Enable mentorship.
4. Personalize.

1. Build a bridge between hiring and onboarding.

To set new hires up for success, tailor every job description to the onboarding plan for that role. Start by asking yourself these questions:

- What are the 30-60-90-day goals for this position?
- What core competencies does someone need to reach these goals?

Use your answers to create an impact-based job description – a job listing that clearly outlines the path to reaching target productivity.

At first, this might feel like more work than you want to invest in a job description. However, impact-based job descriptions are valuable for two main reasons:

- They improve focus on relevant qualifications during interviews.
- They prepare new hires to succeed by communicating clear, consistent expectations.

When developers begin onboarding, use the impact-based job description as your roadmap. This bridge between hiring and onboarding helps to create a smooth transition for new hires. Because they are already familiar with the job description from the interview process, they will feel more confident and prepared to dive into onboarding.

> ***In my experience, you won't see a net boost in team productivity for about six weeks. During that time, re-establish your team norms with all developers, not just new hires. This helps to keep your culture cohesive.***
>
> ***– Brian Leonard***
> *former CTO for TaskRabbit*

2. Overcommunicate.

Overcommunicating is not about bombarding new hires with information. It's about strategically communicating and reinforcing expectations throughout the onboarding process. Here are some tips to get you started:

- Communicate all important information in writing and face-to-face. This is especially important in remote settings, where miscommunications are more likely to occur.
- Establish weekly 1:1s starting on day 1.
- Connect new hires with a peer mentor. More on this in the next section.

What information do new hires need to be successful? Before day 1, document the following:

- Week 1 expectations
- Hardware and software setup
- Important meetings to attend
- Employee Resource Groups that may be helpful
- Company values

Share this documentation with new hires a week in advance of their start date and ask them to review it. Let them know that you will also cover this information face-to-face.points of contact on the team)

During your first 1:1, discuss the following:

- Onboarding roadmap based on your impact-based job description
- Week 1 expectations
- Sources of support when questions arise (documentation and

In this first meeting, it's best to keep the information concrete so that new hires can focus on getting oriented. In week 2, you can have a higher-level conversation about company culture and personal career goals. Share a meeting agenda ahead of time so that new hires can come prepared:

- Company mission and values
- Team mission and working agreements
- Personal career goals

Brian Leonard, the former CTO for TaskRabbit, recommends that you strategically involve your whole team in discussions about working agreements.

3. Enable mentorship.

New developers need the tools to succeed quickly – without slowing down your most productive engineers. Stack Overflow found that for a team of 50 developers, the amount of time spent answering questions adds up to between 278-568 hours of time lost per week across the entire team. That amounts to $12,510-$25,560 in wasted resources.

A consolidated knowledge base can help developers find the answers to many of their questions. However, you can't learn everything from documentation. Effective onboarding connects new hires with dedicated mentors who are equipped to support them.

While trust between managers and developers is crucial, it can feel less intimidating to ask questions of someone who is closer to your level on the career ladder. Having an onboarding buddy or peer mentor ensures that new hires have multiple sources of support. These support structures can help guide developers through their 30-60-90-day onboarding plans and model what your team working agreements look like in practice.

Brian Leonard finds that structured mentorship is especially important for junior developers.
You can plan ahead by:

1. Choosing an experienced developer on your team who is interested in mentorship.
2. Developing a mentor-led project that can orient junior developers to your team's work.

A key goal of mentorship during onboarding is to help developers become increasingly autonomous. However, you don't want to lose this collaborative culture once onboarding ends.

Dr. James Stanier, Director of Engineering at Shopify, advocates for continued mentorship throughout developers' tenure:

"*Going beyond the onboarding phase requires good mentorship and coaching to keep your staff moving in the right direction and increase their impact within their team and the department.*"

The best mentorship plans jumpstart productivity and ensure it continues beyond your 90-day roadmap.

> ***"Helping junior developers become productive is all about planning. When you make an offer to a junior developer, you should already have a mentorship plan in place."***
>
> ***– Brian Leonard***
> *former CTO for TaskRabbit*

4. Personalize.

Every developer enters your company at a different starting point. Joseph Gefroh, VP of Engineering at HealthSherpa, accounts for this by tailoring onboarding to varied learning needs and experience levels.

When developers receive personalized onboarding plans, they don't have to waste their time sifting through resources that don't apply to them. They get the information they need in the most efficient way possible. As a result, you can unlock the benefits of a larger team without slowing down more tenured developers.

You can use tools like custom onboarding to create tailored onboarding plans for your team, even during rapid growth. Custom onboarding enables you to update multiple plans at once and tailor individual plans with low administrative effort. This means that you can maintain a high level of personalization at scale, while keeping your own productivity high.

Growth Plans

Effective onboarding doesn't stop once developers hit their productivity goals. It feeds into growth plans that support developers throughout their careers and keep them around longer. Growth plans enable developers to learn and update critical skills that will accelerate your product and promote their career.

To help your developers meet the interpersonal expectations you set during onboarding, integrate collaboration-based goals into growth plans. These goals should be tailored to each developer's growth areas. For example, they might include:

- Challenging ideas with respect
- Communicating technical tradeoffs efficiently
- Pitching your ideas within the company

This is where your team working agreements and processes become critical. For developers who are working towards more effective collaboration, simply setting a goal of "challenging ideas with respect" is insufficient. It lacks concrete steps to enable success. A more impactful approach is to create a clear roadmap to achieving each goal. Leverage mentorship networks and carve out time during regular 1:1s to provide consistent, actionable feedback on developers' progress. Make the most of these conversations by providing developers with an agenda and clear meeting expectations ahead of time. This enables everyone to engage more fully in discussions.

To maximize the effectiveness of collaboration-based growth plans, implement them as part of a broader culture of learning. This culture should support both technical and interpersonal growth.

Continuous Learning

- **Technical**: developers regularly engage with learning platforms and knowledge bases as needed

  Example: an engineer switching from another team needs to learn your product, tools, and workflows.

- **Collaboration-based**: developers regularly practice upholding working agreements around team communication.

Long-form Learning

- Developers use courses to learn new languages, APIs, and technologies a few times a year.

To build a culture of learning, you need a combination of the right tools and growth-minded management practices. Together, they enable the technical and soft skills that your developers need to work together cohesively.

There is another crucial factor in the success of your learning initiatives: developer motivation. How can you encourage your team to embrace continuous and long-form learning?

Fortunately, the work you've done to build trust and shared purpose will make a huge difference in your team's job satisfaction. Providing additional incentives can also help your team to stay accountable. This might include creating opportunities for devs to share their knowledge, work on exciting projects, and build their portfolios. When you celebrate developer learning and connect it to career outcomes, engineers will be motivated to grow with your product.

How to Measure Success

The management practices we've discussed today will reduce friction on your developer teams and enable more productive working relationships. But how can we measure the impact of these practices over time?

There is no metric that can fully capture developer performance. It is a complex outcome affected by many overlapping factors. While there are numerous frameworks that can help you to navigate this complexity, the prospect of choosing one can be overwhelming. Should you choose a framework that measures output, or one that focuses on interpersonal skills?

To help you make an informed choice, let's cover a few different approaches and how these metrics can support developer growth:

- **DORA Metrics**
- **SPACE**
- **PETALS**
- **Next Steps**

DORA Metrics

DORA is one of the most popular frameworks that DevOps teams use to measure developer productivity. Over seven years, the DevOps Research and Assessment (DORA) team conducted surveys exploring the practices, processes, and capabilities that enable “elite” software developer teams. Based on this research, DORA identified 4 metrics that indicate software delivery performance.

Metric	Description	Elite Performance	High Performance	Medium Performance	Low Performance
Deployment frequency	How often is code deployed to production?	On-demand (multiple deploys per day)	Once per week - once per month	Once per month - once every 6 months	Under once every 6 months
Lead time for changes	How long does it take for committed code to go into production?	Under 1 hour	1 day - 1 week	1 month - 6 months	Over 6 months
Time to restore service	How long does it take to restore service from an interruption?	Under 1 hour	Under 1 day	1 day - 1 week	Over 6 months
Change failure rate	At what rate do deployments cause production failure?	0% - 15%	16% - 30%	16% - 30%	16% - 30%

Despite its popularity, DORA has significant limitations. As an output-focused framework, DORA fails to capture the soft skills that drive productive teams.

The engineering leaders we've spoken with agree: excessive focus on metrics like "deployment frequency" can be hugely counterproductive to team performance. Clement Kao, Product Manager and founder of Product Teacher, discovered this early in his career.

On this particular project, I was a consummate task manager. I gave engineers demands and deadlines but didn't explain the business strategy behind them. Without the context or agency to push back on my specs, engineers built to the letter instead of the spirit of the ticket. My assumptions ruined parts of the architecture, and engineers quickly lost motivation.

My mistake? I failed to invest in the human side of building products together."

– Clement Kao
Product Manager and founder of Product Teacher

Clement learned that this is a common pitfall for early-career software engineering leaders. The solution?

"Build a strong foundation for collaboration."

We know that developer collaboration is vital to productivity. So how can software engineering leaders derive value from an output-focused framework like DORA?

The key is to look at measurement holistically: DORA is one part of the story. By considering DORA metrics alongside more qualitative metrics, you can extract insights about growth-minded management practices.

For example, let's say you have revised your team's working agreements. Since that change, have your DORA metrics shifted? Do these changes align with more qualitative measurements around job satisfaction?

In the next two sections, we will explore more qualitative-focused metrics that can provide helpful context around DORA.

Summary: Why might you implement DORA metrics to measure your team's performance?

- Research shows a clear link between high-delivery performance and the 4 DORA metrics.
- DORA metrics provide a high-level snapshot of the effectiveness of your delivery organization.
- You can pair DORA with more qualitative metrics to track correlations between management practices, team health, and team output.

SPACE

The SPACE framework resists the idea that productivity can be measured by team or individual output. In fact, the lead researcher behind SPACE also helped to develop the 4 DORA metrics – and recognized their limitations.

SPACE presents a multidimensional framework that takes a holistic approach to developer productivity. In addition, these dimensions aren't just for managers: the researchers behind SPACE encourage developers to use the framework to gain insights about their own work.

Summary: Why might you implement the SPACE framework to measure your team's performance?

- SPACE systematically captures a holistic range of factors impacting productivity.
- Tracking SPACE metrics can help you to identify growth areas in your processes and culture.

Dimension	Description	Metrics
Satisfaction and well-being	• How fulfilled do developers feel at work? • Are developers happy and able to practice healthy habits at work?	Self-reported in developer surveys
Performance	What is the outcome of each developer's work?	• Quality of code • Impact on product success
Activity	What is each developer's output?	• Volume of code reviewed • Time spent on call • Number of documents written
Collaboration and communication	How effectively do developers work together and communicate?	• Discoverability of documentation • How quickly work is integrated • Quality of work reviews submitted by developers
Efficiency and flow	To what extent can developers complete work quickly and without interruption?	• Self-reported level of focus • Number of handoffs in a process

PETALS

Si Jobling, an Engineering Manager at ASOS, developed the PETALS framework to track team health in remote environments. While DORA is fully data-driven and SPACE balances qualitative and quantitative metrics, PETALS is based on self-reported surveys from your team.

The model is designed to capture weekly feedback using tools that are already part of your workflow, such as Microsoft Forms or Excel. The feedback is then visualized as color-coded flower petals and used to generate discussion with your team.

Paired with quantitative data, PETALS gives you a fuller picture of your team's health than you can gain through output-based metrics alone.

PETALS measures team health across 5 dimensions:

Dimension	Description	Metrics
Productivity	Do engineers feel they are making meaningful contributions to the team?	Self-reported rating from 1-5
Enjoyment	Do engineers enjoy their work?	Self-reported rating from 1-5
Teamwork	Do engineers collaborate to achieve their goals?	Self-reported rating from 1-5
Learning	Are engineers engaged in work that grows their skills and knowledge?	Self-reported rating from 1-5
Stress	Do engineers experience stress as a result of work?	Self-reported rating from 1-5

How to Measure Success

Here is how you can implement this model for your remote team.

Step 1: Gather feedback

Starting with feedback tools, team members rate each dimension on a scale of 1-5 (e.g., Do you feel you have made meaningful contributions to the team this week? How much have you enjoyed your work this week?). Depending on the team, you can plan for results to be known or anonymous.

Step 2: Visualize feedback

This is where the visuals come in. Your ratings for each dimension become petals, which cluster around your average rating in the center. Each number corresponds to a color from a customizable palette. For easy recognition, you can use a traffic light system (1 is red, 3 is yellow, 5 is green).

Step 3: Share, discuss, improve, repeat

You can provide team members with their individual flowers as well as the flower representing your team's average rating for the week. Over time, this enables you to observe trends and outliers in individual employee and team health. From there, you can discuss the results in retrospectives and make plans to improve outcomes.

Si encourages engineering managers to ask themselves: how might you adapt PETALS to meet the needs of your team? Work with your team to iterate on this framework and tailor it to your context.

Summary: Why might you implement PETALS to measure your team's performance?

- PETALS enables quick, consistent feedback in remote settings.
- Qualitative data can pair with more qualitative frameworks to generate deeper insights.
- The visual element of PETALS will engage your team and spark conversation.

Next Steps

There is no silver bullet framework to improve your developer team's performance. Instead of looking for the perfect framework, select one that makes the most sense for your team's current goals and pain points. Then, collaborate with your team to analyze insights and develop solutions.

Whether you are reviewing self-reported assessments or quantified output, use this data as a starting point to generate conversation. Ask your team:

- How does this data align with your personal experience?
- What factors could be contributing to these results?
- What changes could we make to improve outcomes?

The foundation of psychological safety that you've built will encourage developers to share their thoughts candidly.

When you involve developers in the process of improving outcomes, you make individual and collective growth a priority for your team. This shared buy-in will help to maintain a high-performing mindset as developers work towards technical and interpersonal goals.

GET STARTED: STEPS YOU CAN TAKE TODAY

Get Started: Steps you can take today

As we've explored today, effective collaboration is vital to developer productivity. Working on a team may come more naturally to some developers than others. However, being a "team player" isn't a fixed quality: it is a set of skills that anyone can build with the right support.

How can you help developers build these skills? It is a gradual and iterative process. Let's review the steps you can take to get started:

1. Establish the qualities of highly collaborative teams.

We know that highly collaborative teams tend to have 3 things in common:

Shared Purpose

Establish this by:

- Creating working agreements: documented behaviors that align with your team's values and help you to achieve common business goals.

Trust

Establish this by:

- Building psychological safety: a climate where everyone feels safe to grow from their mistakes.

Clear Processes

Establish these by:
Documenting steps for effective communication in different scenarios. These might include:

- How to navigate disagreement.
- How to deliver and receive feedback.
- How to communicate efficiently in different channels.

2. Consistently apply growth-minded management practices.

Once you establish the qualities of highly collaborative teams, the next step is to help developers meet your expectations. The best way to do this is through consistent application of growth-minded management practices. For example:

- Create a personalized onboarding process that clearly communicates expectations. Connect new hires with multiple sources of support:
  - Documentation
  - Weekly 1:1s
  - Peer mentors

- Create tailored growth plans as an extension of onboarding. These should include goals with actionable steps for improving soft skills.

3. Measure the impact of management practices over time.

To determine the effectiveness of your management practices, you'll need a plan for tracking success. The framework and metrics that you choose will depend on your team's needs. Some frameworks to consider:

- DORA (output-focused)
- SPACE (holistic approach to productivity)
- PETALS (quick, qualitative feedback for remote teams)

Whichever frameworks you choose, remember that no metric can fully capture a team's performance. Use the data to initiate team discussions about your goals, processes, and challenges.

4. Revise management practices to address evolving needs and pain points.

As your company grows and business goals evolve, so will the needs of your team. The management practices that enable success today might not be effective six months from now.

That's why it's important to gather consistent data. Pay attention to changing outcomes so that you know when it's time to adjust your approach. As always, be sure to involve your team in these changes. Developer insights and buy-in are crucial to your success.

We hope these growth-minded management strategies will help you to unlock the full potential of your developer teams.

educative

HAPPY LEARNING!

HOW THE BEST ENGINEERING
LEADERS FOSTER EFFECTIVE COLLABORATION