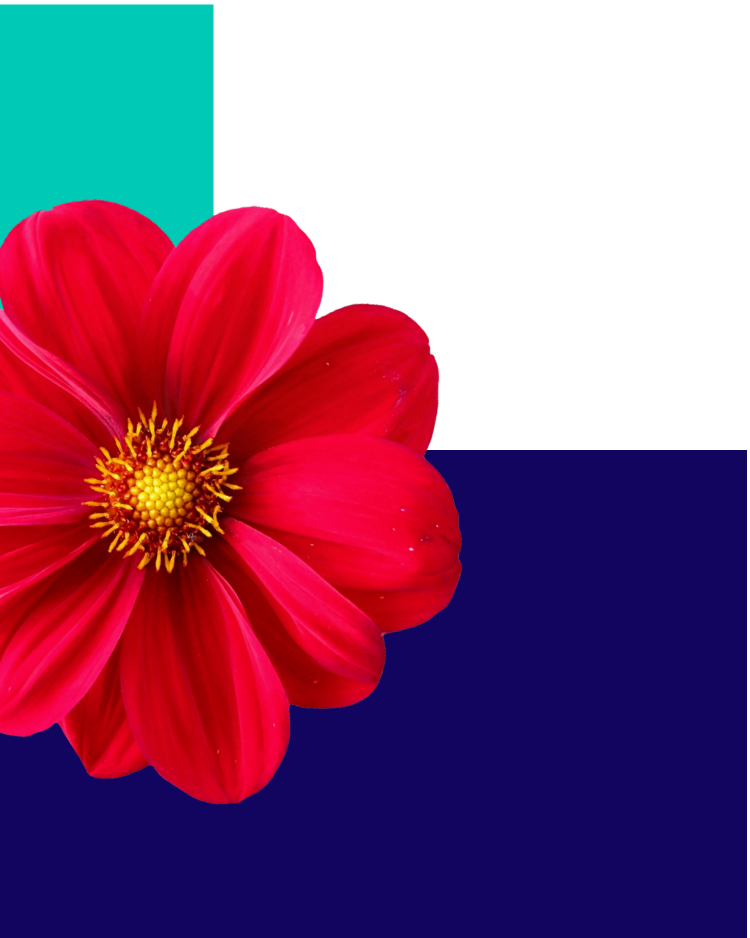
WHITE PAPER 2023

Code smarter, not harder

The DevOps guide to accelerating developer productivity



garden

Intro

> **An engineering manager works more like a gardener** — tending an environment to help their team and products flourish in what will ultimately be unexpected ways.

It's little wonder that business leaders want to maximize developer productivity. Talented development teams bring tremendous value to an organization. With developers in short supply and looming economic uncertainty, getting more out of existing teams is a competitive advantage.

The pitfall is that developers aren't machines. You can't just turn a dial and crank them into high gear. Treating developers like gears in a machine leads to toxic practices (we're looking at you, developer ranking chart) that ultimately hurt productivity.

Great development and engineering teams aren't built; they're grown. Unlike building a bridge or a skyscraper where you can predict exactly what's needed and use materials that behave in a predictable way, software and development teams evolve and change in response to their environment.

We **build** static things; we **grow** things that we want to evolve. And when it comes to improving developer productivity, the investment is well worth it. McKinsey & Company found that the savings produced by reducing each developer's wasted time by five minutes on a team of 500 developers could support a full team of developers working on standardization.

In Garden's first survey of more than 400 developers, commissioned from an independent research firm, we dug deep into the developer experience to identify where time is wasted and what frustrations developers face.

In this paper, we look at solutions DevOps and engineering managers can implement at the enterprise level to reduce friction and increase developer productivity.

We'll explore where productivity is lost and how engineering managers can weed those problems out to provide the right conditions for productivity to grow.

It's time to get rid of things that are throwing shade on your developers' day-to-day work. By making more time for the type of work that makes them happy (think: more coding, less of everything else) you'll see their productivity multiply.

TABLE OF CONTENTS

01 How more complex systems impact productivity

02 Putting productivity metrics in perspective

03 Where is productivity lost and what’s the cost?

04 What would happen if developers had more time?

05 Digging deeper into the developer experience and productivity

06 Four ways to cultivate happier, more productive developers

07 The rise of platform engineering and productivity tools

08 Increase your team’s productivity now with Garden

How more complex systems impact productivity

The proliferation of cloud-native capabilities offers tantalizing possibilities for building containerized architecture, unique digital experiences, near-instant data processing, AI and ML applications. Developers are experimenting with and adopting new tech at a breakneck pace. This fuels innovation, but it also creates complex systems that evolve organically — and this can get a little messy.

In a recent study on developer velocity at work, McKinsey & Company found that companies use as many as 50 different tools. And these tools can be extremely complicated: Kubernetes is barely a decade old and developers who've mastered it are in limited supply. Yet 70% of IT leaders said they work for organizations that use Kubernetes, according to Red Hat's State of Enterprise Open Source Report.

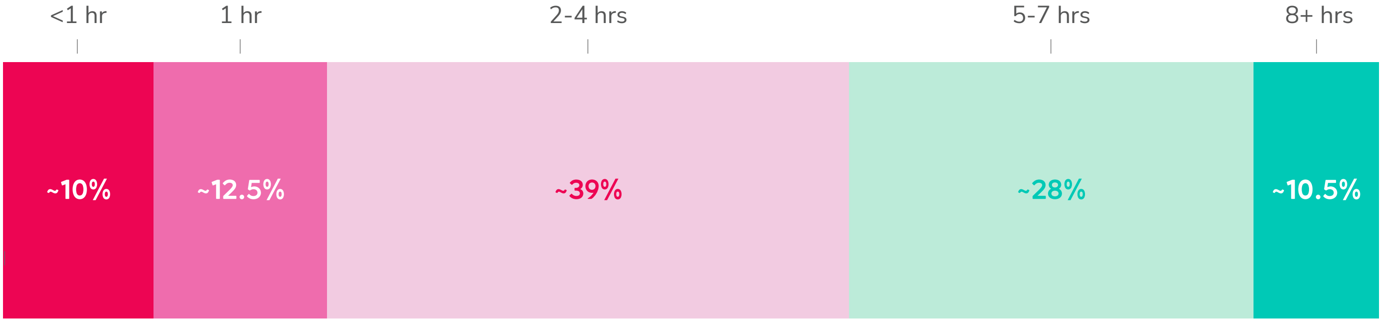
> “Application complexity is an issue that most organizations need to deal with, and how you deal with it will impact the future of your application, the health and stability of your development organization, and the future of your business.
>
> — Lee Atchison, InfoWorld, “A cure for complexity in software development”

More time wrangling tools means less time coding

We adopt new tools because they offer exciting possibilities, but those benefits come at a cost. Garden found that developers spend 14 to 16 hours every week wrangling internal tools, setting up environments, and waiting for tests, builds, and pipelines. Developers at organizations using Kubernetes were at the higher end of that range.

More time spent wrangling tools means less time coding. Similar to Garden's findings, ActiveState observed an approximate 20% decrease in time spent programming between their 2018 and 2019 surveys. The number of respondents who spend eight or more hours a day programming dropped by almost 50% in the same timeframe.

How many hours per day do developers spend on programming?



Most developers reported spending four hours a day or less programming.
Source: ActiveState, Developer Survey 2019

Putting productivity metrics in perspective

It's hard to talk about productivity without metrics, but developer metrics are often associated with toxic practices. Let's be clear: productivity metrics should be used to measure impact or identify areas for improvement — not to spur developers to code harder and faster. We're pretty sure business leaders and customers care more about business outcomes, such as more frequent deployments and fewer incidents and rollbacks, than they do about lines of code.

Healthy vs. harmful use of metrics

No business can simply "metric" their way to sustainable, higher productivity. Metrics help organizations, teams and individuals figure out what they're doing well, where they can improve and how changes impact performance. They help managers understand how different parts of the process are working, predict timelines and identify blockers.

> **"The goal of tracking and analyzing software metrics is to determine the quality of the current product or process, improve that quality and predict the quality once the software development project is complete.**
>
> — Alexandra Altvater, Stackify, "What Are Software Metrics and How Can You Track Them?"

While metrics are helpful, they can be misused and they rarely tell the whole story. Focusing too narrowly on one metric, or one type of metric, can be misleading and detrimental to improvement efforts. A common example of this is measuring lines of code. This can inadvertently reward developers who write a lot of low-quality or simplistic code, and punish those who accomplish the same thing more efficiently.

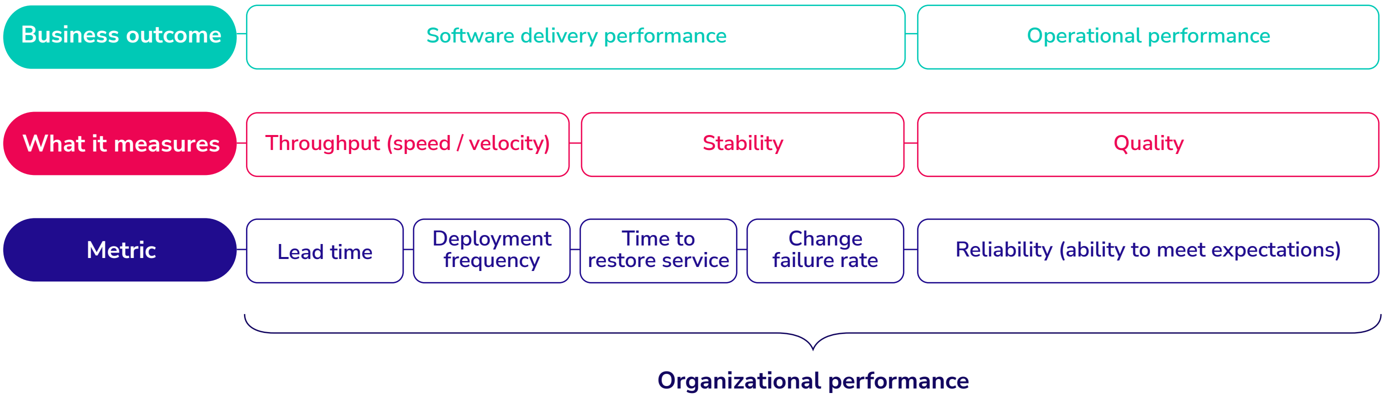
We want developers to write elegant code that works well and is easy to maintain. We want high-quality software. Focusing on a narrow measure like lines of code can result in a sort of busy work that artificially mimics productivity, but fails to result in better outcomes.

Choosing the right metrics

No one seems to agree on one golden metric that works for every business. A good practice is to choose some metrics that focus on speed and velocity and some that focus on quality and stability. This prevents unintentional trade-offs.

Google Cloud's DevOps Research and Assessment (DORA) team identified five metrics that strike this balance. These metrics are based on eight years of research and measure both software delivery performance and operational performance. Teams that strike this balance and excel in all five measures exhibit exceptional organizational performance.

DORA's 2022 State of DevOps report highlights the importance of measuring different aspects of performance. "We have evidence that suggests that delivery performance can be detrimental to organizational performance if not paired with strong operational performance," write the authors to explain why they added a fifth metric: reliability.



How individual metrics can ladder up to a balanced scorecard: speed and velocity are weighed with quality and stability to prevent unintentional trade-offs.
Source: 2022 Accelerate State of DevOps Report

Metrics within the metrics

While the DORA metrics are a valuable tool in identifying areas for improvement in DevOps, developers are often more focused on the early part of the process. Meanwhile, engineering managers are focused on the full cycle: What's the delta between when a developer thinks they're done (they've written and delivered code) and when they're actually done (code has passed all the tests or is in production)? How long does each step take?

Measuring how the early part of the process is working can help identify areas for improvement that can lower change failure, increase deployment frequency and lower lead time for changes. These metrics can include time between starting a new branch and getting a pull request merged, time between starting new work and getting the test to pass, and how frequently you need to push to get code done and working.

Looking at underlying metrics that impact high-level metrics can help teams improve productivity by making small changes in the right places.

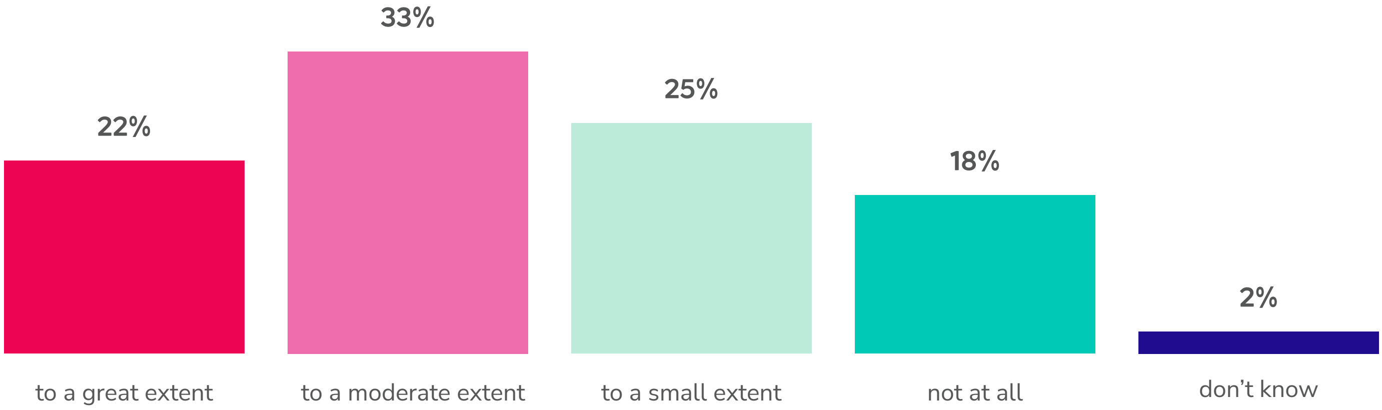
> “**Instead of managers challenging teams to deploy more frequently (and leaving them wondering how), you can identify and remove specific roadblocks to reduce friction.**

Then celebrate the results.

Where is productivity lost and what's the cost?

Increasing productivity is, in part, about reducing time wasted. We're not talking about long bathroom breaks or sharing a funny TikTok video with coworkers. We're talking about waste that's become an inherent part of the process. Process inefficiencies might seem small or be overlooked as "business as usual," but they frustrate developers and add up to hours of wasted time.

More than half of engineers reported delays due to inefficient processes:



Source: Haystack, Study to understand the impact of COVID-19 on Software Engineers

Waiting is the hardest part

Garden estimated that more than $61 billion in developer productivity and value is lost due to inefficiencies in the software development pipeline in this post about our earlier research.

We can use the same method to calculate costs in terms of wasted time for individual businesses and teams:

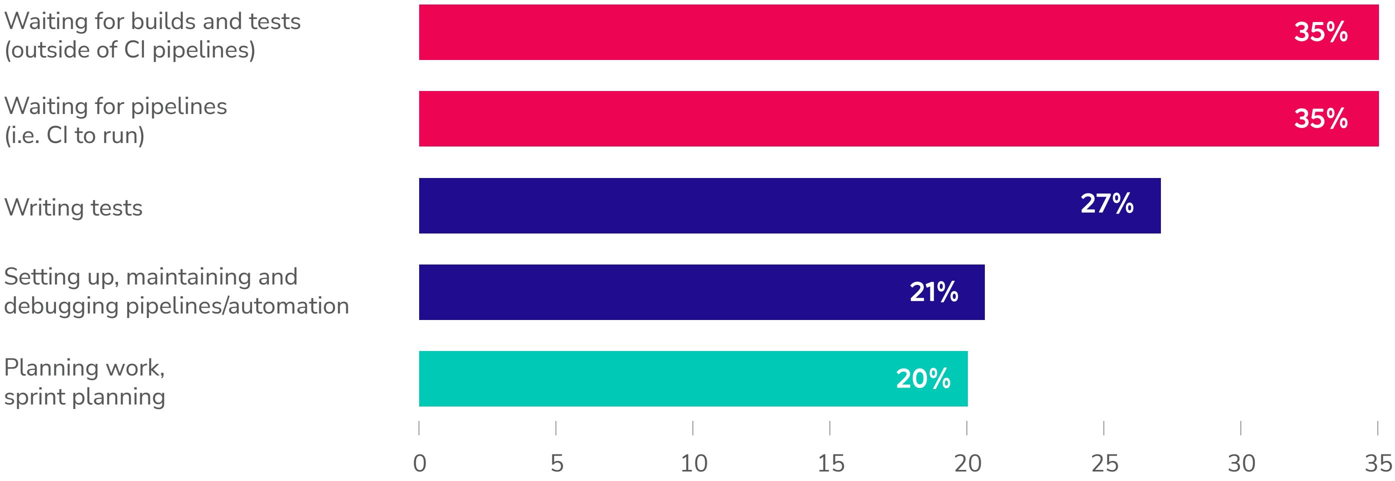
- The 2021 median salary from the US Bureau of Labor Statistics for software developers, quality assurance analysts and testers was $109,020 annually, equivalent to $52.41 per hour.
- McKinsey uses 500 developers as an example team size.
- Formula: [hours wasted per week] x $52.41 x [52 weeks per year] = annual value of wasted time

Applied here, we can see that a team of developers typically spends 4.9 to 6.3 hours each week waiting. That's a cumulative 2,450 to 3,150 hours each week — at a cost of $7–9 million per year.

For our last whitepaper, we surveyed 400 developers and DevOps team members. We asked which tasks they considered to be time wasted. Waiting for builds and tests outside CI pipelines (35%) and waiting for pipelines to run (35%) were at the top of the list.

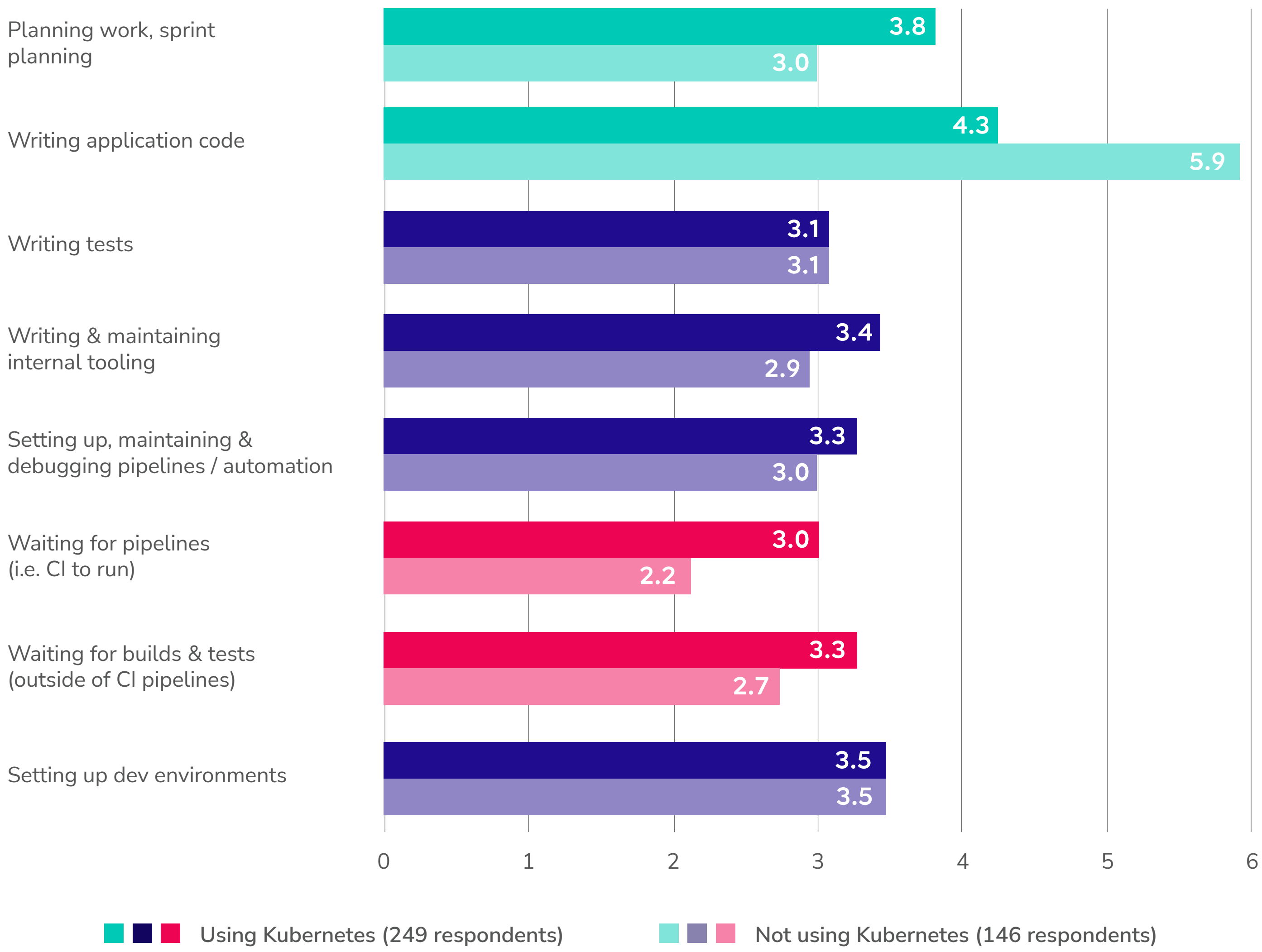
> “Developers are tired of throwing things over the wall and waiting for a response.

Which of the tasks do you consider to be time wasted? Top 5 responses



Source: Garden original research

How many hours per week are spent on the following tasks? (on average)



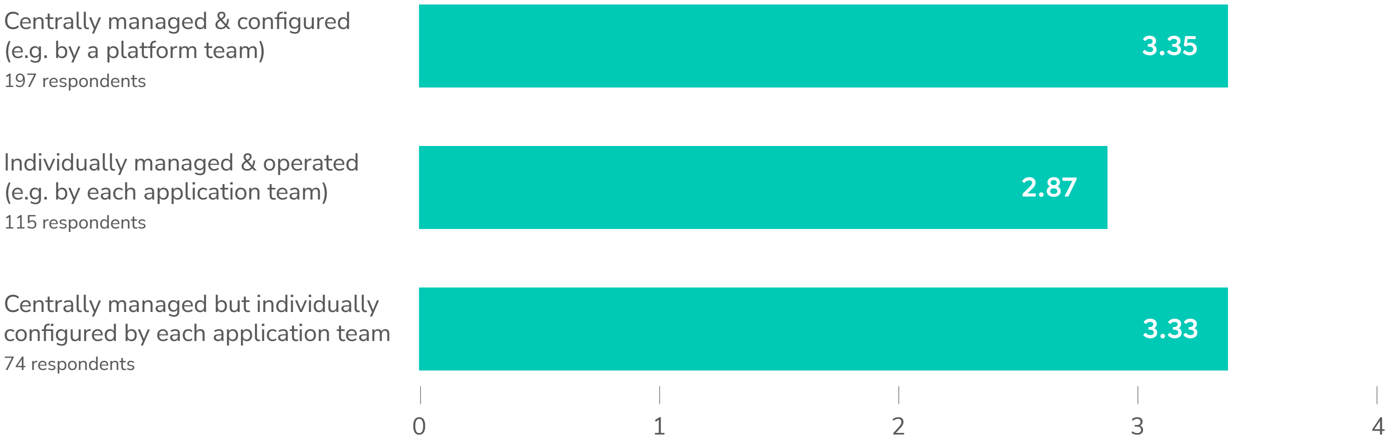
Developers don't just spend their time on productive work, such as planning and writing code. They're also stuck waiting for pipelines, builds and tests (in pink). And there are a number of requirements (in purple), such as writing tests, maintaining internal tooling, debugging and setting up dev environments, that can be automated or substantially time-reduced with better tooling, such as Garden.

Managing complex systems takes too much time

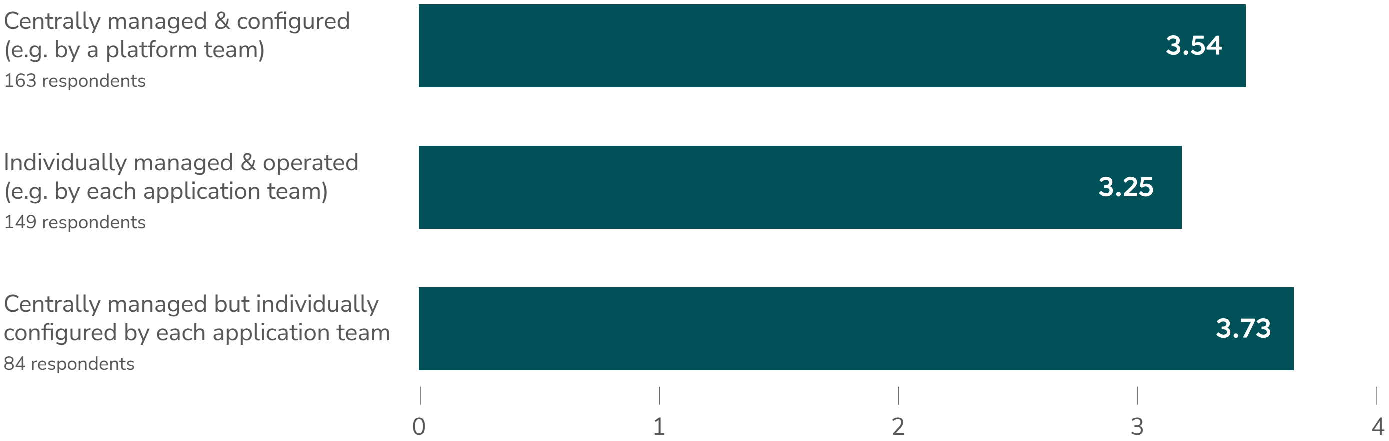
The time needed to manage complex systems surpasses time spent coding for many developers. In an ActiveState developer survey, 62% of developers said they spent part to all of their time managing dependencies, and 61% reported spending part to all of their time building a library or package.

This is a symptom of trying to apply the same practices that worked ten years ago to more complex systems. Developers are spending valuable time acting as the glue between systems and trying to set up the environments they need.

How many hours per week are spent setting up, maintaining and debugging pipelines / automation if CI / CD is...



How many hours per week are spent setting up dev environments if dev / testing environments are...



Source: Garden original research

We've found that it's possible to save three to four hours per developer on these tasks each week. For a team of 200 developers, three hours saved weekly is more than 30,000 hours annually — equivalent to gaining the productivity of 15 extra developers, or more than $1.7 million per year.

The cost of context switching is higher than you think

Once developers have shipped code off to CI/CD, they're obliged to wait ... and during that downtime, they're typically hopping between different tools to find something else to do. This context-switching takes up far more time and mental bandwidth than most people imagine.

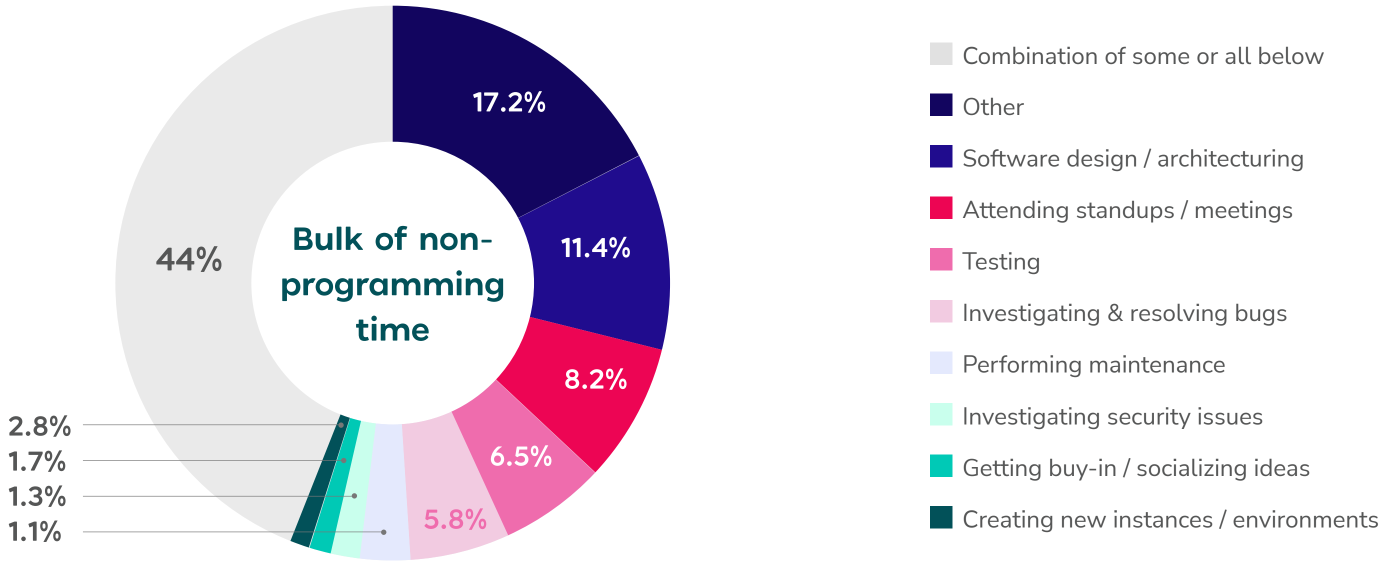
In a study of 20 teams at Fortune 500 companies, people took a little over two seconds to switch applications, but they switched applications almost 1,200 times a day, Harvard Business Review reported. That adds up to just under four hours a week. And that's just the physical time needed to switch.

The math on this also paints a picture of wait and waste: 4 hours of context switching for each of 500 developers will cost a company $5 million per year in lost productivity — not to mention lost revenue, as the wait time further delays the product's time to market.

In addition to the almost four hours people spend switching applications, we also need to consider the time it takes to refocus on a task. Researchers at Georgia Tech looked at this specifically in developers. In most sessions, it took developers several minutes to make their first edit after resuming a task, but in 30% of the sessions this "edit lag" was over 30 minutes long.

Developers have a lot more than programming on their plate

Developers spend a lot of time on non-programming tasks. While some of these tasks are no doubt necessary, being more cognizant and intentional about non-programming time can help teams make small, but valuable, gains in productivity.



Source: Developer Survey 2019 - Open Source Runtime Pains - ActiveState

What would happen if developers had more time?

While it's possible to put a dollar value on lost productivity, the real value is in what developers would do with more time. So what would developers do if you freed them from all the waiting and unnecessarily complex system management tasks?

We asked them:

49%

of respondents would develop new products & services

46%

would improve speed & delivery of existing products & services

44%

would improve security for existing products and services

Digging deeper into the developer experience and productivity

> **Putting DX at the center of their efforts can help organizations improve employee attraction and retention, enhance security and quality, and increase developer productivity.**
>
> McKinsey & Company, "Why your IT organization should prioritize developer experience"

It's easy to make the connection between how developers spend their time and productivity. Freeing developers to spend more time on more productive tasks equals more productivity. But there's another component at play here: developer happiness.

Researchers at the University of Oxford found that happy workers are 13% more productive compared to discontent colleagues working the same number of hours. Forbes furthered this research with a focus on developers and found that happy developers are 1.8 times more likely to deploy to production multiple times a day compared to their grumpier counterparts. Now that's a nice way to improve your deployment frequency!

What's more, Forbes found that

> **happy devs in a mature devops organization are 32% more likely to recommend their company to others.**

More productive teams are happier

> **“High-performing software engineering teams deliver 53% better outcomes in employee experience and productivity compared with low-performing teams.**
>
> 2020 Gartner Software Engineering Team Effectiveness Survey

It turns out the things that make developers happy make engineering managers and companies happy too. Over 92% of developers in mature DevOps organizations showed high levels of job satisfaction, according to Forbes. Compare that to only 61% of developers in immature DevOps practices.

Developers are happier when they feel more productive. In fact, feeling unproductive at work was the number one (45%) cause of unhappiness among developers — even above salary, according to a StackOverflow survey on developer happiness.

When we asked developers what’s frustrating them at work, it wasn’t surprising to see that factors related to productivity (or a lack thereof) were high on the list. Developers know their time is valuable and it doesn’t feel good to spend that time waiting for feedback, pipelines and builds.

Which daily tasks contribute to developer frustration?

→ Which of the following cause the most frustration in your job?

55% Communication between teams & functional groups is difficult

54% Slow feedback loops during the development process

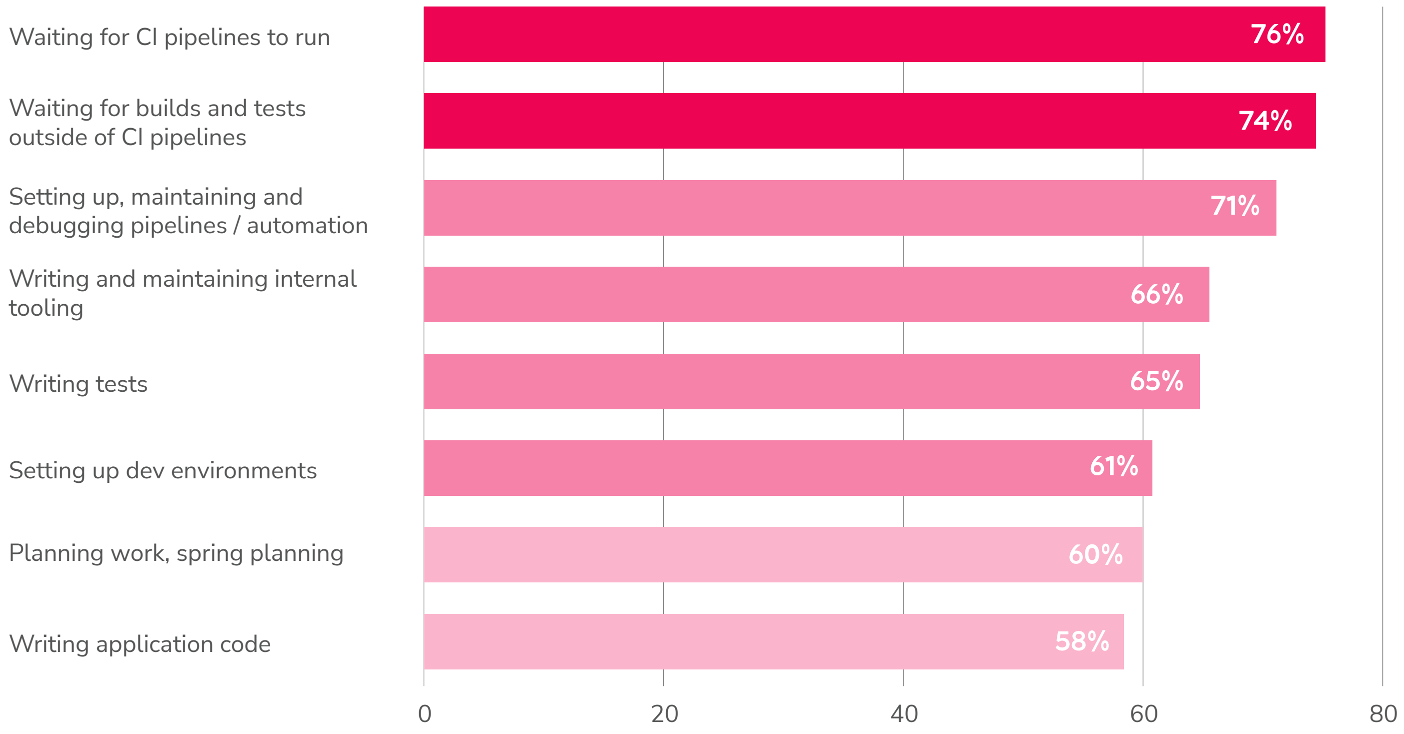
49% Lack of flexibility in choosing tools

46% Solving solved problems (building tools which don't add additional value to our core business)

44% Inadequate tooling and processes to support remote work

40% Lack of direction from upper management on strategic priorities

→ Percentage of respondents frustrated by daily tasks



Source: Garden original research

Four ways to cultivate happier, more productive developers

1

Remove roadblocks

Analysts at Gartner and McKinsey both recommend a “servant-leadership” approach that focuses on removing roadblocks and empowering teams to be successful. “When leaders identify and resolve roadblocks, for example, their teams are 16% more effective. Likewise, when leaders take on coordination with stakeholders like project managers or governance partners, they up team effectiveness by another 11%,” writes Laura Starita, in Garner’s “3 Ways to Make Your Software Engineering Team 50% More Effective”.

2

Empower developers

Developers know where their time is wasted and can be valuable partners in creating better standards and processes. Involving developers in standard setting makes them 23% more effective than their counterparts who don’t participate in standard setting, wrote Starita.

3

Make work a happy place (psychological safety)

Toxic metrics and pushing developers to perform better without improving the process or their experience are unfortunately still problems in the industry. Creating a safe and happy workplace is critical to helping developers do their best work and be more productive. The most important cultural attribute is psychological safety, according to research on developer velocity by McKinsey & Company. This means protecting developers’ ability to experiment and fail, and investing in tools and systems that minimize the cost of those failures.

4

Invest in best-in-class tools

McKinsey & Company identified best-in-class tools as the top contributor to business success — enabling greater productivity, visibility, and coordination. Yet only 5% of executives ranked tools as one of their top-three software enablers. “The underinvestment in tools across the development life cycle is one reason so many companies struggle with “black box” issues,” write researchers in “Developer Velocity: How software excellence fuels business performance.”

The rise of platform engineering and productivity tools

Gartner expects that by 2026, 80% of software engineering organizations will establish platform teams. These teams might be called different things — DevOps, developer experience, developer tooling or developer enablement teams — but their purpose is similar. They're tasked with optimizing the software delivery process by reducing complexity and simplifying the way developers access and use the tools they need.

These specialized teams focus on creating and maintaining an operating platform — a tool or collection of tools that sit between developers and the tools they need. The goal is to create self-service capabilities that reduce friction and overhead for developers.

> **“For the organization, such platforms encourage consistency and efficiency. For the developer, they provide a welcome relief from the management of delivery pipelines and low-level infrastructure.**
>
> Gartner 2022: What Is Platform Engineering, and What Does It Do?

Build vs. buy

As with any tool that could potentially be built in-house, there's a build vs. buy debate around developer platforms.

Realistically, companies with complex setups or lots of legacy and bespoke tools aren't going to be able to buy something that does everything they need. On the other hand, building from scratch can take years. One big box retailer with a team of 3,000 developers told us they spent the last three years building internal tooling for platform engineering … and they're still building.

Instead of spending years to create software that's not part of the core business purpose (the retailer, for example, needs to focus on developing the digital experience for its end customers, not on the developer experience), platform engineers can buy key components

> **“This reduces the surface area of what they need to build and maintain in-house, allowing them to focus on components that differentiate the business.**

Industry experts echo this advice:

- "Each platform team needs to focus on the needs of the company and avoid building in-house alternatives to tools that can be easily bought. Rather than focusing your energy on generic needs that can be met using pre-existing products, think about what your differentiator is within the industry." 8 Ways to Build Platform Engineering Teams | Fellow.app

- "It doesn't matter if your homegrown CI/CD solution is superior today, commercial vendors will catch up eventually. Platform teams should always ask what is their differentiator. Instead of building in-house alternatives to a CI system or a metrics dashboard and compete against businesses that have 20 or 50 times their capacity, they should focus on the specific needs of their organization and tailor off-the-shelf solutions to their requirements." Gartner 2022: What Is Platform Engineering, and What Does It Do?

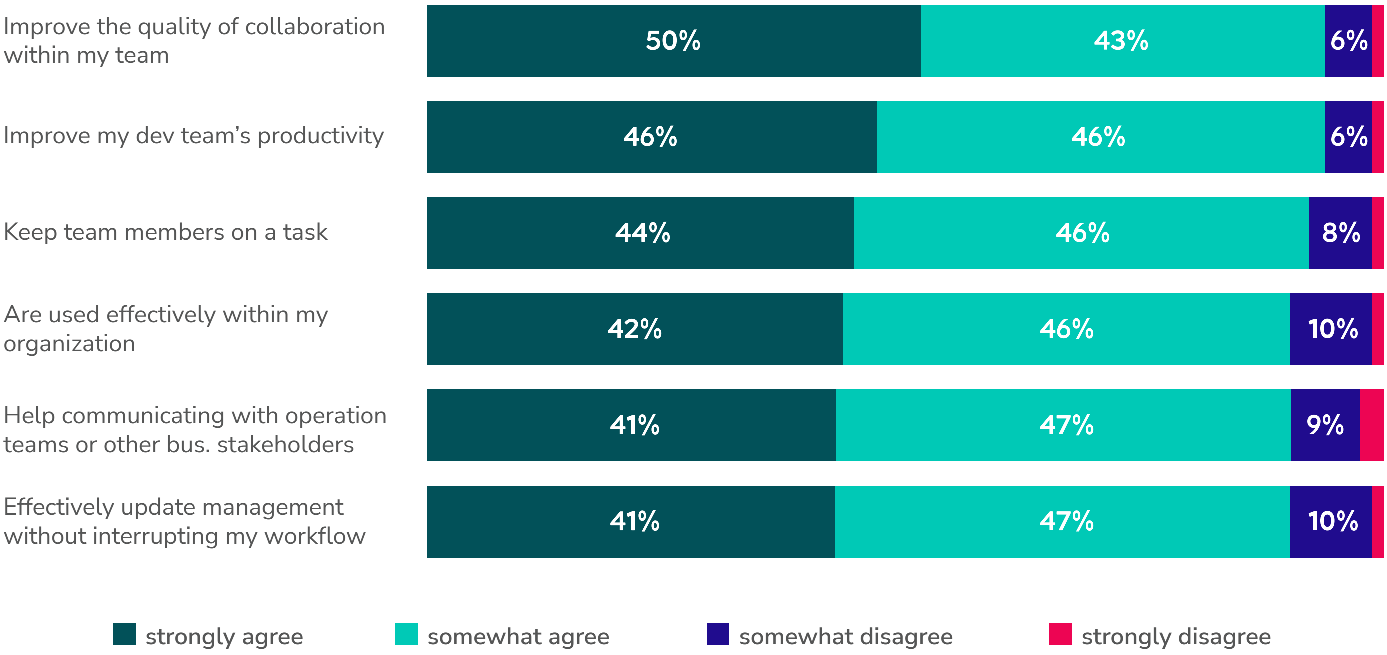
Given the value of even small gains in productivity, tools that help developers do more will pay for themselves. Buying tools enables you to deliver benefits to your organization sooner. No one wants to spend years building the perfect platform — only to have a competitor buy the same capabilities off the shelf.

Do productivity tools work?

We're so used to the way things work (or don't work) now, it can be hard to imagine them functioning differently. But productivity tools do work. They improve collaboration, organization and communication. They alert management to blockers and reduce those little friction points that leave developers feeling frustrated and slow processes down.

Consider these stats from [Zenhub's 2022 Software Developer Happiness Report]

"Productivity tools..."



Developers positively rate many aspects of productivity tools, including 88% who agree that the tools help them update management without interrupting workflow.
Source: Zenhub's 2022 Software Developer Happiness Report

Increase your team's productivity now with Garden

You can invest in a large team and spend a year or more building your own platform to reduce developer frustration and increase productivity. Or you can get started with Garden today.

Garden.io was named a 2022 Gartner "Cool Vendor in Platform Engineering for Improving Developer Experience," in a report authored by analysts Manjunath Bhat, Arun Chandrasekaran, and Stephen White. And it's been adopted by Fortune 500 brands and customers known for innovation and a great developer experience. In fact, one Garden customer increased its internal developer Net Promoter Score (NPS) by 50% in nine months.

Designed by developers, Garden eliminates the time-consuming back-and-forth between tools, reduces friction and improves the developer experience. By stitching the tools your team needs together, we create a central hub that supports CI/CD and automation.

With Garden, you can keep the tools you like, including internal tools. We help you connect and make better use of the tools you already invest in, so your team can build, deploy, run and test smoothly.

Garden provides the insights engineering managers need

Engineering managers need timely data to fix friction points and bottlenecks before they become a pain point for developers. And they want insights that help them forecast speed of delivery and keep projects on track.

Simply by using Garden, developers generate a ton of neatly structured data. Garden's Stack Analytics and Stack Streams give you access to this data, both in real-time and in aggregate. You can see logs for everything going on in your project; understand the flows between all your builds, deployments and tests; and get more actionable information at every turn.

We'd love to show how easily Garden can integrate with your existing stack and alleviate pain points for you and your developers.

LET'S SCHEDULE SOME TIME TO TALK →

In the meantime, show your developers some love and invite them to use Garden's free tool built for developers by developers. If they use it and love it, they'll have you to thank.

garden