2020 DIFFBLUE

DevOps And Testing Report

How can organizations build a better DevOps testing culture?

diffblue
AI for Code

Introduction

With **DevOps adoption on the rise** year after year, achieving DevOps is a goal for most, but a reality for few. Organizations with the best intentions still find their software delivery lifecycle pipelines **getting clogged by inadequate testing**—a key part of DevOps that, in practice, tends to be an afterthought.

Just like **insufficient culture is often a barrier to adopting DevOps**, an inadequate testing culture can be a barrier to adopting the good testing processes that facilitate DevOps. The growing popularity of movements like Test-Driven Development (TDD) reflects the acknowledgment that increasing testing often requires drastically changing behavior: Organizations and individuals have recognized that in order to encourage people to do something they typically find excuses to avoid—e.g., write unit tests—new norms have to be established on a large scale.

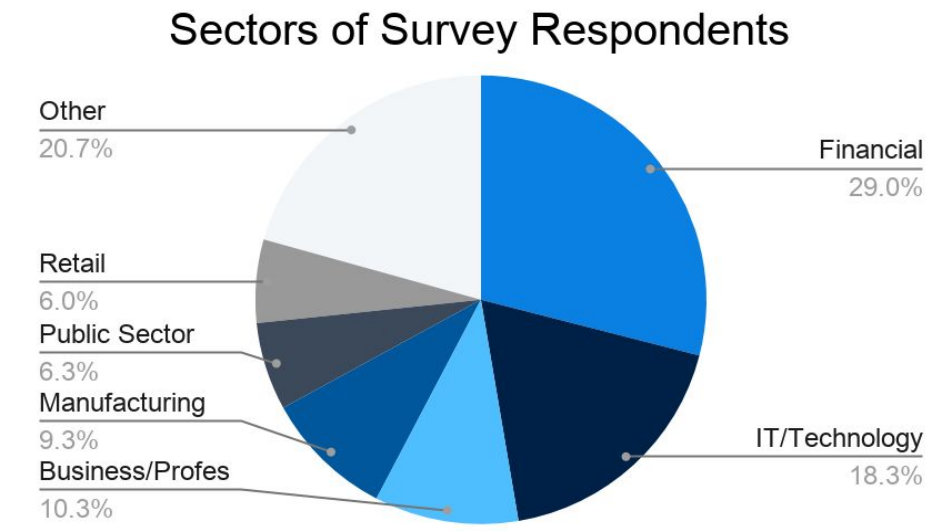
But TDD is a time-intensive process that is not realistic for many organizations, and other behavioral solutions to the problem of inadequate testing (such as implementing code coverage targets) haven't solved the core problems. Without discussing what testing culture actually looks like, or what facilitates it, it won't improve—and neither will the issues surrounding the rest of the DevOps pipeline, which relies on well-tested code. So what can be done to support behavioral change and build a culture that actively supports testing?

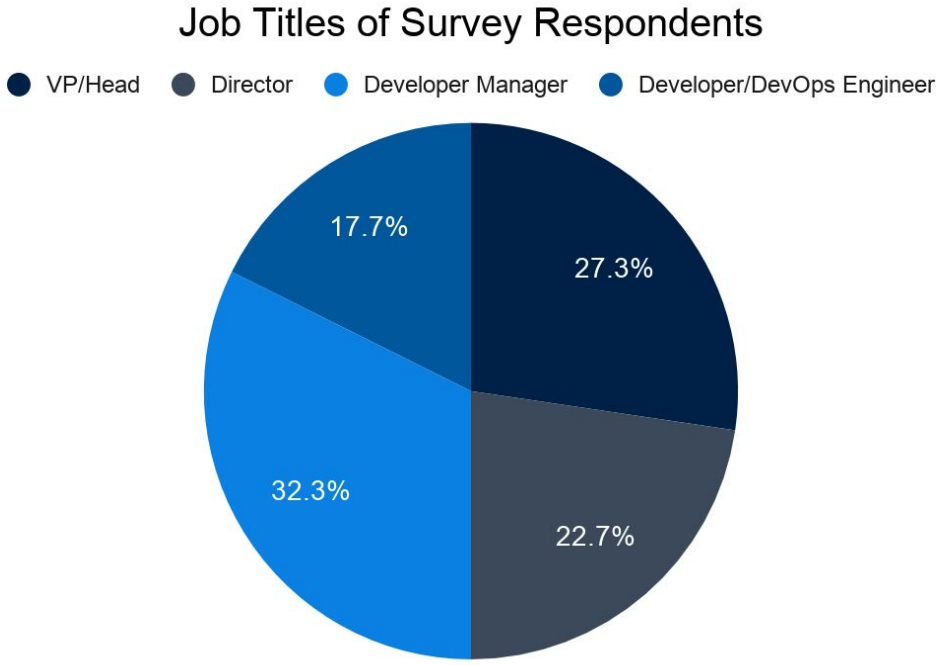
To answer this question, we surveyed 300 developers, DevOps professionals and senior decision makers in companies that have at least partially adopted DevOps about their experiences with testing as part of DevOps, and their DevOps challenges more broadly.

Methodology and Demographics

For this study, 300 respondents were surveyed online by the research agency **Vanson Bourne** in the spring of 2020. The survey included a mixture of Likert scales, multiple choice, and open-ended responses. Of the respondents, 200 were from the US and 100 were from the UK; all worked in software development, application development and DevOps at companies with at least 500 employees.

Sectors included financial services, IT/Technology, business and professional services, retail and various others. Participants all had job titles that fell in one of two seniority levels: developers/DevOps engineers and their direct managers, and senior management (VP, Head or Director level roles).


Sectors of Survey Respondents
Other
20.7%
Financial
29.0%
Retail
6.0%
Public Sector
6.3%
Manufacturing
9.3%
Business/Profes
10.3%
IT/Technology
18.3%


Job Titles of Survey Respondents
VP/Head
Director
Developer Manager
Developer/DevOps Engineer
17.7%
27.3%
32.3%
22.7%

Key Findings

The findings from this study reinforced the importance of having a culture supportive of testing as a central theme. But more specifically, while challenges vary based on the size of the company respondents work in, common problems with DevOps often fall along a linear path that starts with misunderstandings between the developers who work with code and senior management, and ends in a testing bottleneck.

Typically, that process looks like this:

1. **Differing opinions between developers and management** about the quality of the company's code and the efficiency of its SDLC lead to...
2. **Insufficient time and resources** dedicated to testing from management, which leads to...
3. **Ineffective adoption** of resource-intensive solutions to testing problems, which leads to...
4. **Manual testing** as a key bottleneck in a CI/CD pipeline

This report breaks down the findings in each of these stages, and introduces a new maturity model we've developed based on the results—the DevOps Testing Culture Maturity Model—to identify how organizations can start building a better DevOps testing culture.

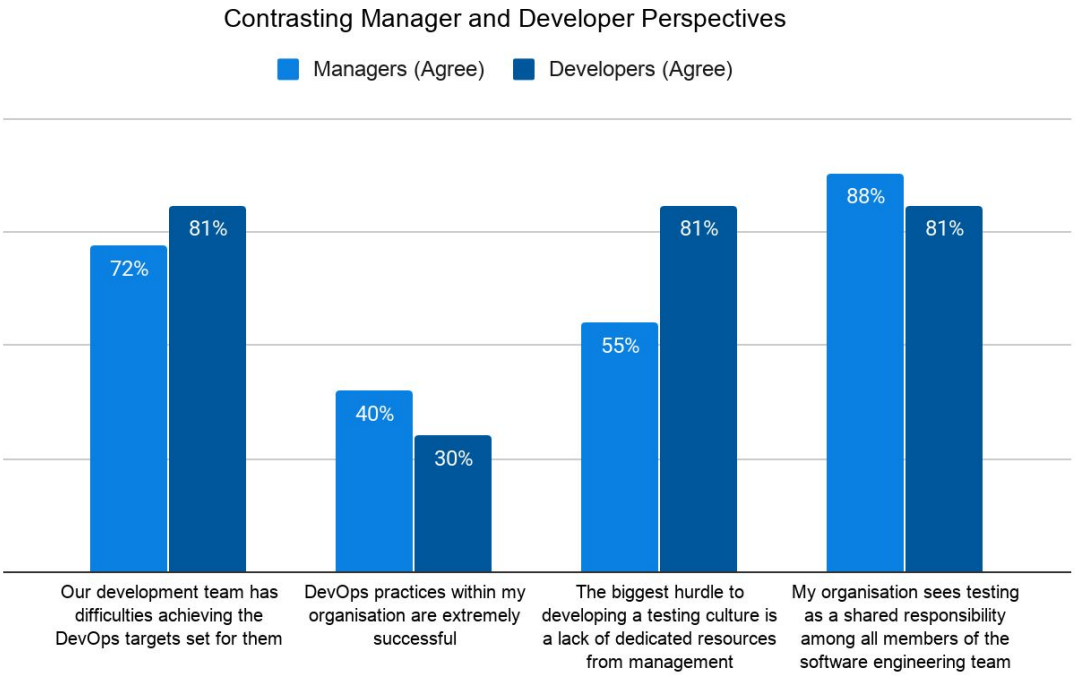
1. The disconnect between developers and senior management

One of the most consistent patterns in responses was a difference of opinions between engineers and senior management. Typically, this played out with senior management slightly overestimating the efficiency and efficacy of their teams' testing and DevOps processes, and underestimating the support developers needed from above.

81% of software developers, for example, said that they have difficulty achieving the DevOps targets set for them, while only 72% of senior managers thought their teams had any difficulties in this area. Similarly, 40% of managers said their organization's DevOps practices are extremely successful, while only 30% of developers said the same. Managers were also slightly more likely (88%) than developers (81%) to think testing is seen as a shared responsibility among all members of the software engineering team.

One of the starkest differences was in response to the phrase "The biggest hurdle to developing a testing culture is a lack of dedicated resources from

management:" 81% of developers agreed, compared to only 55% of managers.



The divide between the perceptions of developers and managers can add up to an inaccurate "big picture" view of the state of software development in the company—one in which management is unaware of the reality of the challenges faced by their developers, and therefore might not be giving them the tools or support they need to succeed, even when testing is causing real problems (as supported by the findings in part 2).

2. Resources are not invested in testing

One of the core issues facing testing as a practice is that it is not quick or easy to do; it requires an investment of time, budget, or (most often) both. But getting that investment can be a challenge. As stated above, 81% of the developers who responded to this survey agreed that the biggest hurdle to developing a testing culture is lack of dedicated resources from management. When asked which stage of the DevOps pipeline respondents feel their organization places as its top priority, 51% chose developing, with deploying in second place (24%). Testing fell in last place (11%).

Next to an increased budget to allocate to new tools and training, time is one of the most limited resources in software development, and deciding which stages of the SDLC to set aside time for reflects the value the organization places on that stage. However, even among the relatively DevOps-mature organizations at which our respondents are employed, only 35% said time for testing is always built into their organization's release schedules. The majority of respondents (57%) said testing time was only built in most of the time.

Investing time in establishing formal processes is another indication of the importance a process holds, but only half of the respondents said their companies have fully adopted formal code-testing processes.

Similarly, hiring skilled developers to do certain tasks is another indication that those tasks are important to the business. 39% of developers strongly agreed with the statement, "A shortage of skilled developers causes delays and lowers the quality of the software my organization produces," but only 25% of managers said the same, suggesting an underlying disagreement about the quality of code that can be expected when an organization has a shortage of developers, and how important it is to hire a large enough team.

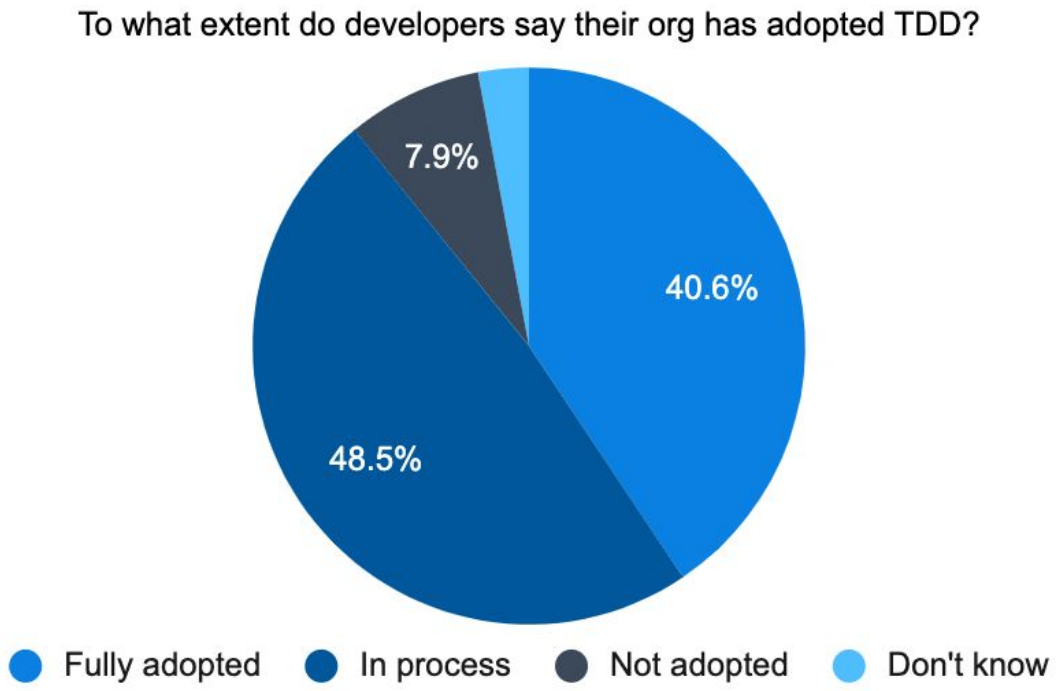
3. Organizations are not effectively adopting high-resource testing solutions

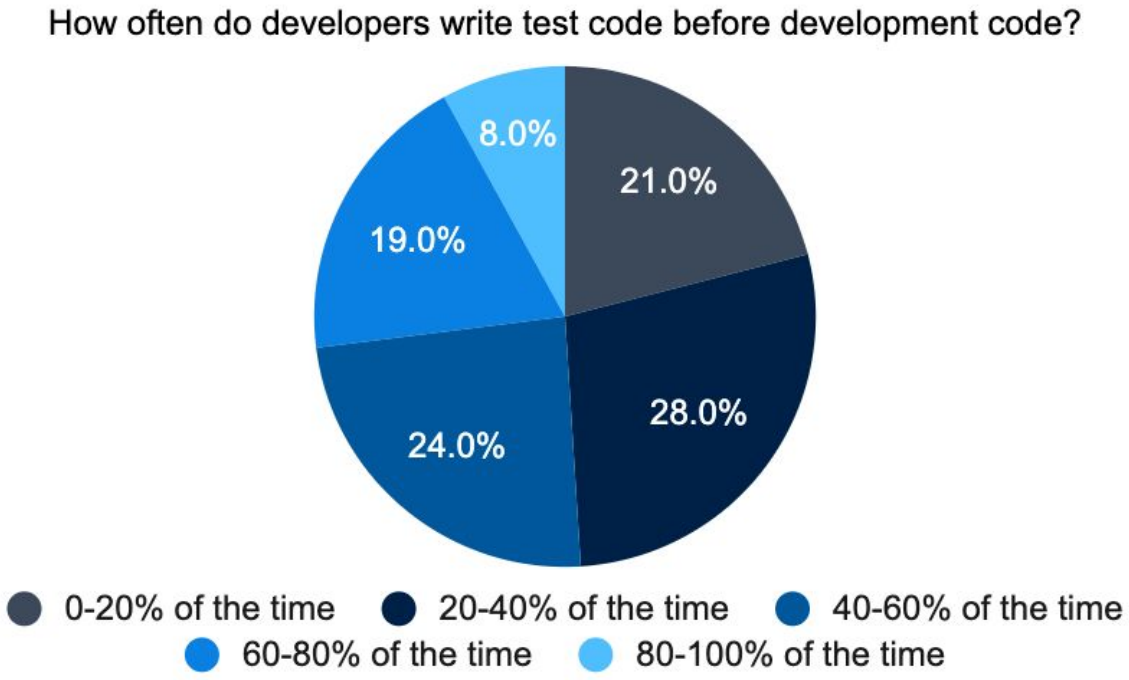
If testing were an inherently enjoyable activity, extra resources might not be required for developers to find time to write more tests. But as the results of the survey showed, most developers are not internally motivated to spend their limited time on testing. 55% of responses to the question "Which factors are the most influential on your motivation for writing unit tests?" were extrinsic factors (e.g. "I have code coverage targets to hit") compared to intrinsic ones (e.g. "I believe unit tests help us achieve our DevOps goals").

Without increased investment in making testing easier or more efficient, it is not surprising that resource-intensive solutions are not being implemented—and realistically, there are no purely behavior-based solutions out there today that don't also require a heavy investment of time or budget.

Test-driven development, for example, is a time-intensive process that requires developers to write tests for code before writing the code itself. It's embraced as a way to maintain high code coverage and write better code, and 41% of the developers who responded to the survey said their organizations have fully adopted TDD. However, when asked the same

question in a slightly different way ("How often do you write tests before writing the code under test?"—the definition of TDD) only 8% of developers said they do this at least 80% of the time.



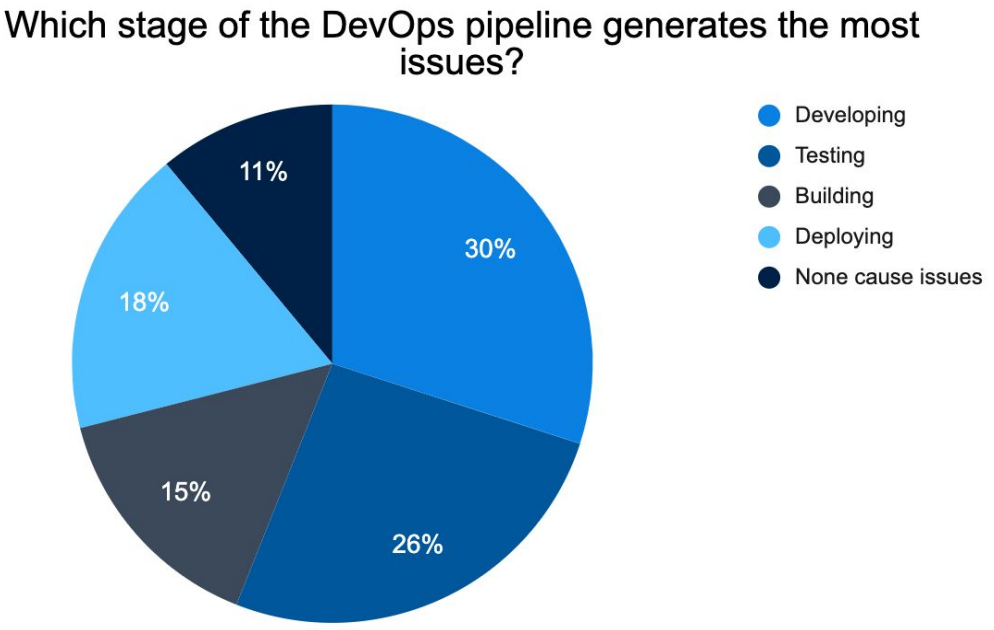


It's possible that this disparity suggests a disagreement about what the practice of TDD actually entails, but it could also indicate that many organizations view TDD adoption as a goal, but don't have the means to actually practice it. In either case, the outcome is the same: many organizations say they have adopted TDD, but few actually have.

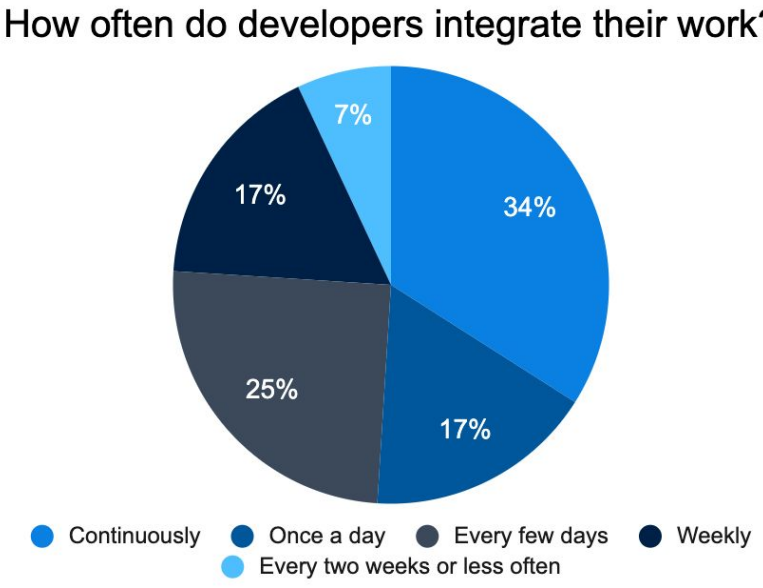
4. The Manual Testing Bottleneck

To avoid introducing bugs, code has to be well tested and of generally good quality before it's integrated. Code quality issues caused by insufficient testing have the potential to slow down the entire DevOps pipeline and delay the release cycle (and 91% of respondents agreed with this statement).

For the participants in the study, code quality seems to be an issue preventing other DevOps-related initiatives from being successful. 78% of respondents agreed that their organization's regression test suite could use improvement, and 77% said the same of their organization's software quality. 26% of respondents said testing is the stage that causes their organization the most issues within its DevOps pipeline—second only to developing (30%). Unsurprisingly, 89% of survey respondents said finding a solution for testing bottlenecks is a priority.



Perhaps as a result, 63% of respondents to this study have not yet achieved CI, and 85% have not yet achieved Continuous Deployment, suggesting that the above issues related to testing are holding back their overall DevOps success.



The Testing Culture Maturity Model

After analyzing these findings, we have developed a Testing Culture Maturity Model that takes into account the wider culture surrounding testing in organizations, especially as it relates to DevOps, including:

- Seeing testing as a shared responsibility among all members of a team
- Holding teammates accountable for testing
- Increasing feedback between developers and senior management
- Building in time for testing, rather than seeing it as an afterthought
- Investing resources in testing by allocating time and budget to actively experimenting with new tools and methods and evaluating the results
- Embracing automation

Organizations can be split into four levels:

- **Beginners:** Organizations at this level have no testing culture or formal procedures; any testing that occurs is ad hoc and driven by individuals.
- **Late adopters**: These organizations are late adopters of testing tools and methods; they may engage in some formal testing, but it is not prioritized or invested in.
- **Early adopters**: An early adopter is among the first wave to pick up new testing tools and methods; the organization is quick to find and follow new best practices, and devotes some resources to regular improvement.
- **Cultural leaders**: Cultural leaders have organization-level adoption of testing practices, which results in the company leading the pack with new techniques, pioneering the newest tools, and developing a continuously evolving culture of testing. Feedback about what's working (or not working) is collected and reviewed regularly.

Ideally, this model will be useful for organizations seeking to "level up" their testing and get (or stay) ahead of the competition. To find out how your organization's testing culture measures up, **take the DevOps Testing Culture Maturity Assessment.**

Discussion and Final Takeaways

Simply having testing practices in place is not the same as having a culture conducive to testing. Culture is about more than just behavior; it also includes the norms a group follows and the common tools they use.

When it comes to software tools in particular, the SDLC has advanced unevenly: tooling has simplified and accelerated some processes—like connecting software with plugins, or packaging and shipping applications—but the remaining bottlenecks typically require the kind of work that is much harder to automate, e.g. writing code.

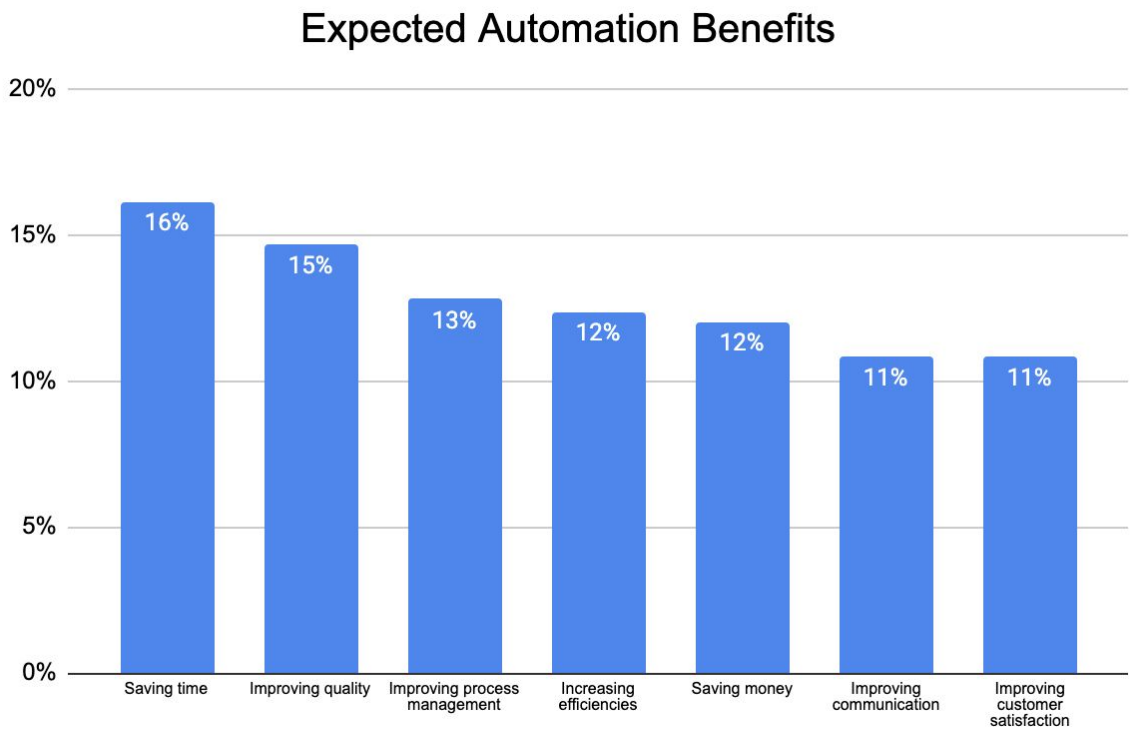
This is why many companies have taken a behavioral response to solving the testing problem, such as implementing TDD and code coverage targets. But as the findings above indicate, without fundamentally facilitating the process of testing itself, these behavioral approaches simply don't work. At best, a company will have to start investing more in time-intensive processes like TDD, which isn't always a realistic option. At worst, individuals will report that they've adopted solutions without actually changing anything about their behavior, therefore leaving the problems with testing unresolved.

All of the participants in this study work at organizations that have already actively started to adopt DevOps. When asked about their own company's testing culture, most respondents self-reported that their organizations were early adopters or leaders in terms of testing (86%). If this is accurate, it does not bode well for the overall state of testing. As long as there is minimal investment in the time and tools to facilitate testing, behavioral solutions on their own will suffer from low uptake. The piece that has long been missing, but that's finally beginning to be addressed, is investing in tools that can fundamentally improve the testing process itself while reinforcing behavioral best practices.

Changing how testing works with the right tools

New AI solutions are here for the harder-to-automate aspects of testing—like the creation of tests themselves—and the expectation is that they will make the testing process more manageable and efficient: 90% of survey respondents agreed that they expect to use AI to improve productivity by 2025.

The top expected benefits of automating more processes (e.g. the creation of code) were saving the organization time (16%) and improving the quality of the work (15%). 86% of participants agreed with the statement, "Being able to automatically create test code would eliminate a bottleneck in the testing stage."



What's next for DevOps?

Introducing cultural changes that facilitate testing as a whole—such as increasing communication between development and management, supplementing manual labor with new AI tools, and ensuring that testing a shared responsibility for all—will make it possible to eliminate the testing bottleneck at the source and finally achieve wider DevOps goals.

To stay up-to-date with suggestions for improving testing culture in your organization, check out **our blog** for regular articles on key DevOps topics. For more Diffblue research, you can also read our **2019 developer survey**, where we asked developers how they would improve software speed, quality and cost.

About Diffblue

Diffblue is changing the way code is developed. Diffblue Cover uses AI for Code to write unit tests that help software development teams and organizations efficiently improve their code coverage and quality. Headquartered in Oxford, Diffblue is funded by Goldman Sachs and Oxford Sciences Innovation.

Learn more and get a free trial of our tool Diffblue Cover at **diffblue.com**.