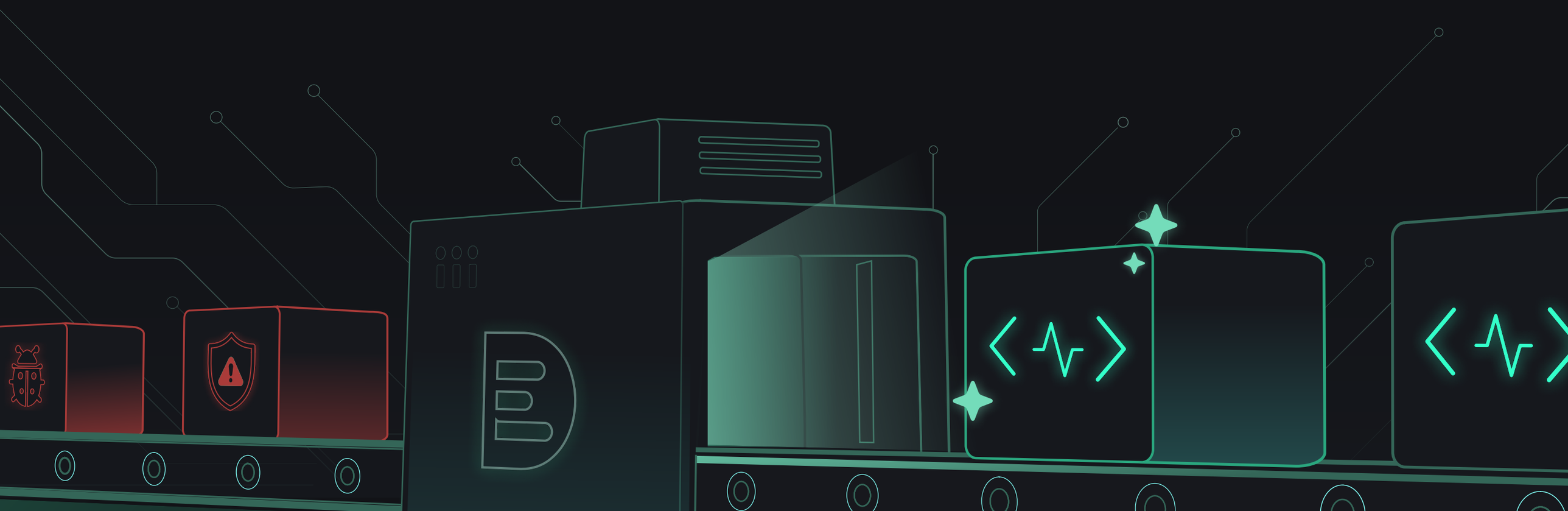
deepsource

State of Code Health

How engineering and security leaders approach code quality and security in 2023.

2023



Preface

Code health refers to the state of an organization's codebase regarding its quality, maintainability, and security [1]. It's an abstract measure of how well-engineered the code is and how easy it is to maintain and extend over time. By focusing on code health, engineering leaders can ensure their software is reliable, efficient, and secure, ultimately leading to better business outcomes.

In this whitepaper, we present the findings from our survey of software developers and application security specialists in the technology industry on improving the process and outcomes of maintaining code health.

[1] "Code Health: Google's Internal Code Quality Efforts." 3 Apr. 2017, https://testing.googleblog.com/2017/04/code-health-googles-internal-code.html. Accessed 1 Feb. 2023.

Contents

Introduction 1

The Survey 3

Code Health: Processes 6

Code Health: Outcomes 20

Conclusion 24

About DeepSource 25

Introduction

Promoting code health improves developer experience and productivity as they deliver products or features with shorter iteration time, less development effort, more stability, and enhanced performance. Besides faster and cost-effective software delivery, ensuring code health helps engineering leaders remediate technical debt.

The underlying ethos is that if you want to go fast, meet schedules, and keep your customers and managers happy, you should keep your code as clean as possible. Still, a 2018 poll found the worldwide cost of dealing with “bad code” to be as high as $85 billion per year[2]. Those numbers only address costs in terms of development time and payroll; the costs of shipping defective software or being beaten to market by a competitor are also worth considering. Since the effects of lousy code persist until its removal, the only suitable approach is to deal with it early and regularly.

In the build-up to 2023, the tech industry has scaled back or even eliminated entire teams to cut costs. The sector layoffs and cost-cutting measures now require tech leaders to become proficient in achieving more results with fewer resources. Engineering leaders are facing the challenge of optimizing the efficiency of their teams: cutting delivery costs and elevating developer productivity.

According to the Association of Computer Machinery (ACM), technical debt wastes 23-42% of developers’ time[3] . Gartner predicts that Infrastructure and Operations leaders actively managing and reducing technical debt will achieve 50% faster delivery times by 2023[4]. That aligns with a McKinsey study that showed companies that manage their tech debt effectively could allow engineers to spend up to 50% more time on work that supports their business goals[5].

Besides preventing teams from working extra hours, efficient code health measures reduce the cost of shipping a product or feature and later improvements. A product with good code quality often requires fewer rounds of testing since all the minor issues are quickly discovered and rectified.

[2] "Software developers are now more valuable to companies than money." 6 Sep. 2018, https://www.cnbc.com/2018/09/06/companies-worry-more-about-access-to-software-developers-than-capital.html. Accessed 1 Feb. 2023

[3] "Code Red: the Business Impact of Code Quality - InfoQ." 17 Oct. 2022, https://www.infoq.com/articles/business-impact-code-quality/. Accessed 1 Feb. 2023.

[4] "Assessing Technical Debt to Prioritize Modernization Investments." https://www.gartner.com/en/publications/how-to-assess-infrastructure-technical-debt-to-prioritize-legacy-mmodernization-investments. Accessed 1 Feb. 2023.

[5] " Tech debt: Reclaiming tech equity - McKinsey." 6 Oct. 2020, https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/tech-debt-reclaiming-tech-equity. Accessed 1 Feb. 2023.

The Survey

The survey had two sections: the *process* and the *results* of ensuring code health.

Process

- How often code quality and security is analyzed
- Tools used to track and improve code health
- Considerations when measuring code health

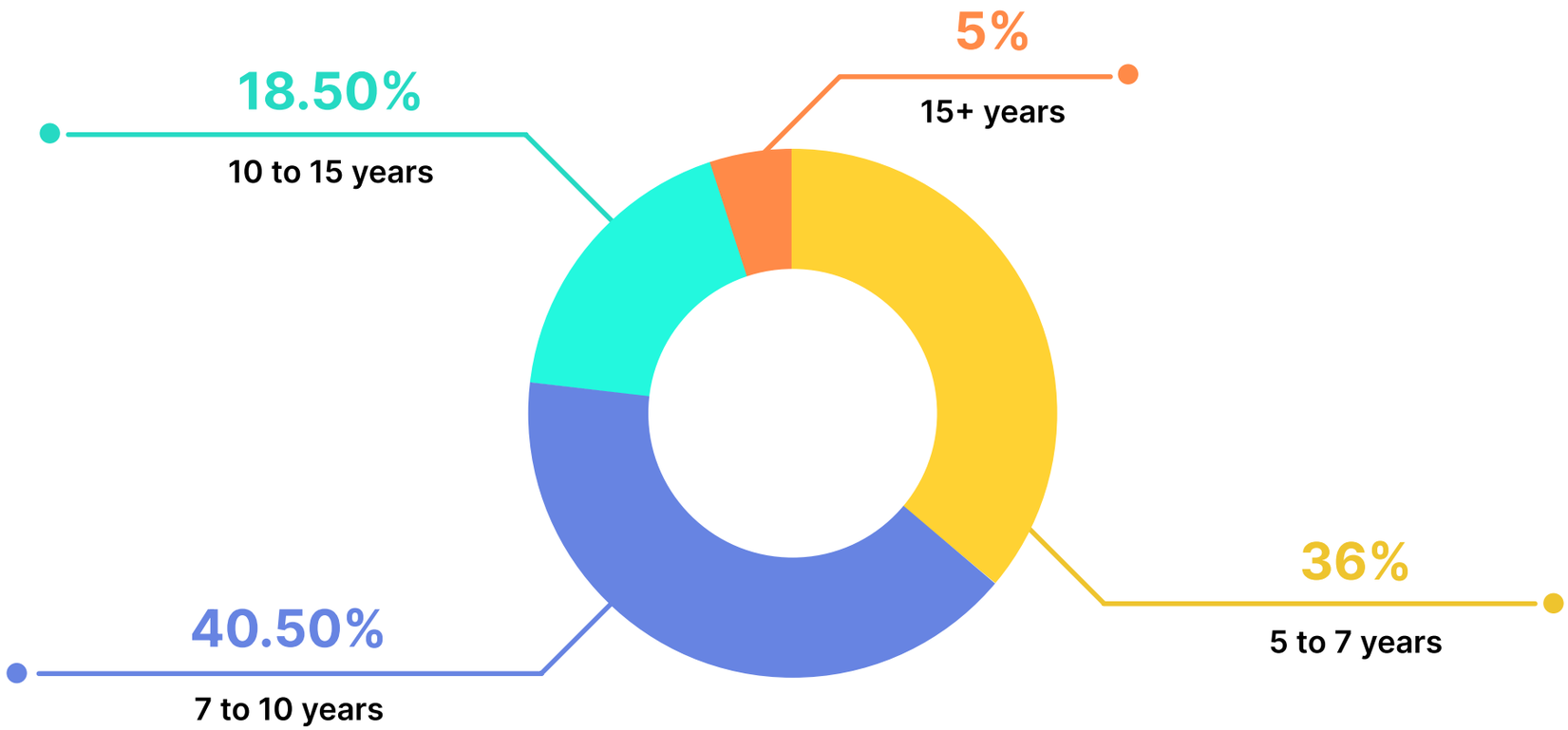
Results

- Overall satisfaction with their code health
- Alignment vis-a-vis determining severity and priority of code health issues
- Perception on the impact of the processes on onboarding

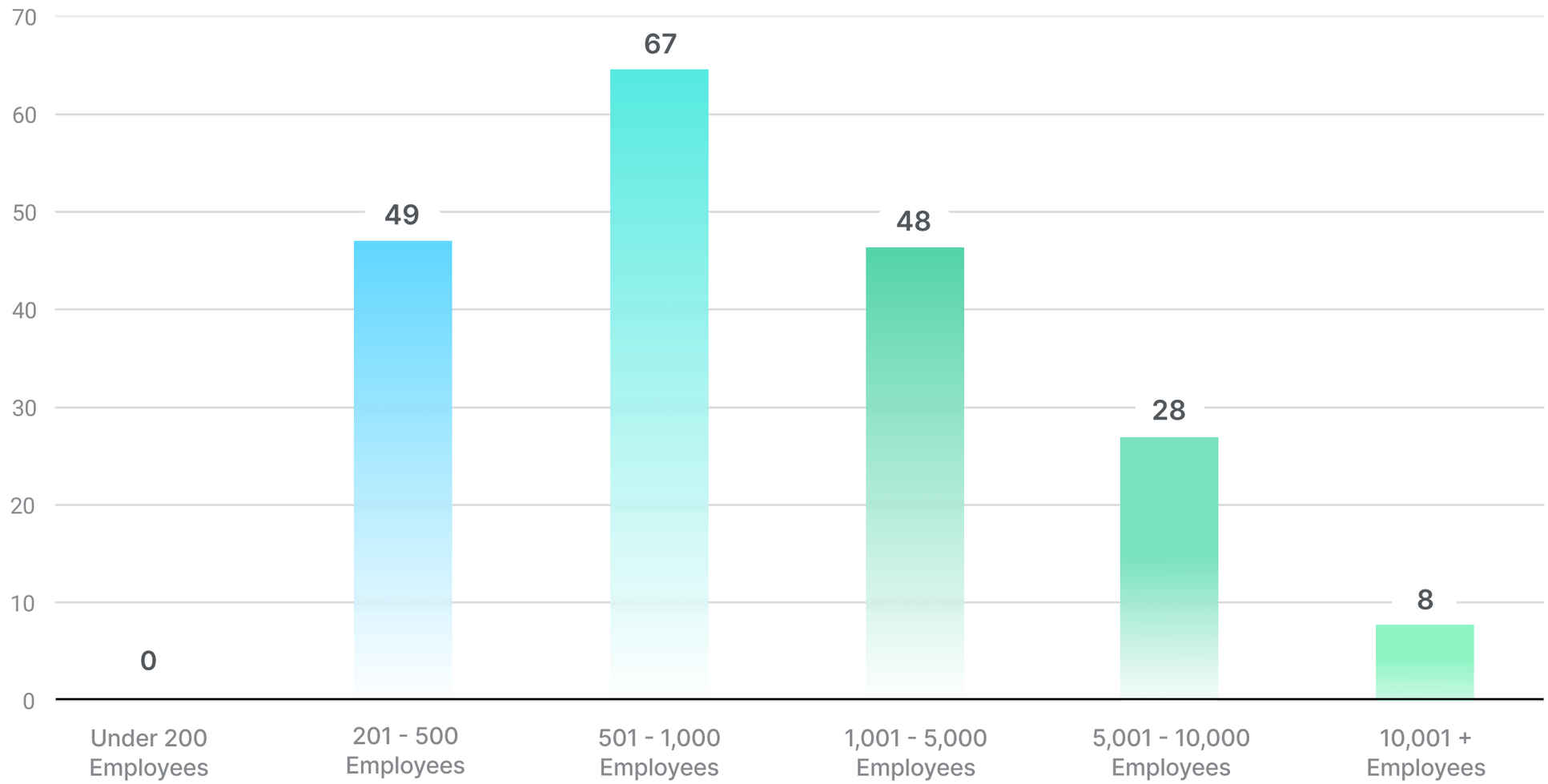
Survey Audience

We surveyed professionals currently working in software development or application security, with at least five years of experience in those roles. They are based in the United States and work at companies with at least 200 employees.

For how long have you been part of software development or software security (including previous companies)?



What's the size of your current company?



SECTION ONE

Code Health: Processes

Respondents at smaller companies (200-500 employees) are less likely to run code analysis continuously and were more likely to do so daily (28.57% of the respondents in that category). At least 34% of all survey respondents are continuously analyzing their codebase. Nearly 60% are running continuous analysis or during every release.

How often do you (or does your team) analyze the quality and security of your code?

Challenges in the Code Review Process

Conducted to find bugs and improve the overall quality of the software, the code review process stands out as a tried and tested method in an ample palette of applications that enables the systematic examination of software source code. Since the review process is an additional step between writing and shipping code, it might impact your team's ability to remain efficient. Some ways the code review process halts developer productivity include longer shipping time, reduced focus on other tasks, and longer review times for extensive reviews.

Optimizing the Code Review Process

Automation can improve the code review process by streamlining repetitive tasks, reducing errors, and increasing the speed and efficiency of the review process. Teams use automated tools to check coding standards, detect potential bugs, and perform static code analysis. Automated tools can provide suggestions for improvement and allow multiple reviewers to collaborate and track changes in real-time. That helps reduce the time and effort required for code review, allowing developers to focus on more critical tasks.

Automated tooling effectively enforces code standards, identifies vulnerabilities, tracks key metrics, and gathers files. Still, teams should avoid relying entirely on tooling and forgoing team member involvement to conduct code reviews. Instead, they should view those tools as an extension of code reviews and a way to enhance the process.

What is the Code Quality Process Supposed to Look Like Moving Forward?

Numerous organizations have varying code quality objectives, significantly influencing the nature of their security processes. Reasons why the security process often intersects with code review efforts include:

- Code review can help usher in some test processes' automation, saving time and manual effort by specifying project-related rules for the test process.

- Code review helps identify errors in the code early, fixing issues as they are detected.

- Code review improves source code quality and security since data leaks and other security issues are reported more often. It can also identify vulnerabilities in runtime, depending on which type of code analysis your testing team implements. Additionally, it improves the coding standards among developers.

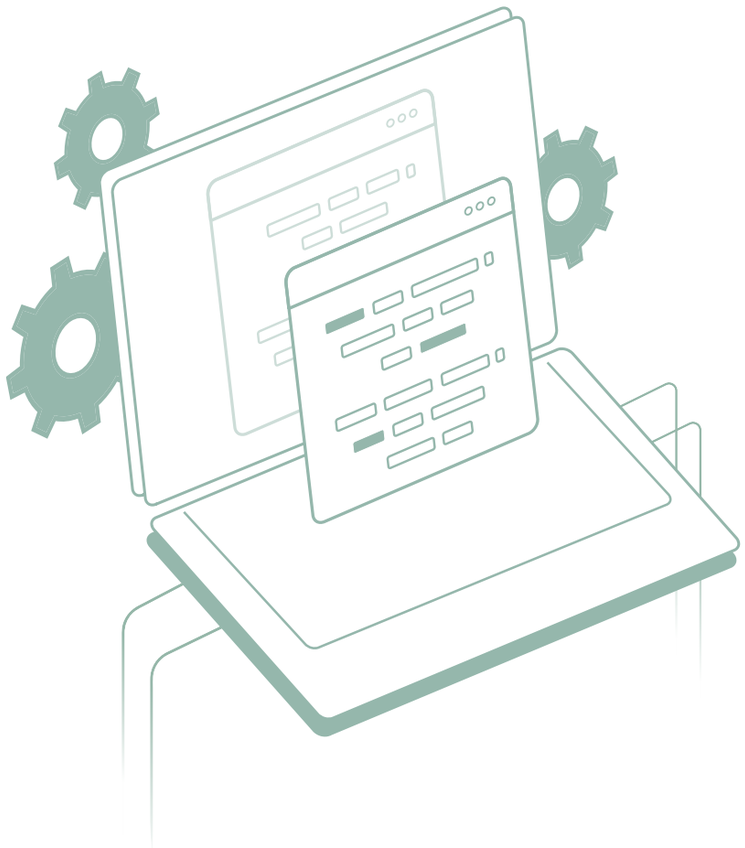
- Code review ensures faster time-to-market, enabling companies to test the code faster and simultaneously release that product to cope with the market demand.

- Code review enables seamless integrations with different development life cycle phases, allowing companies worldwide to adopt Agile practices for increasingly faster releases.

Employing Static Analysis

The method involves analyzing the source code for potential issues without actually executing it. When used during the code review process, several code quality and security issues can be automatically detected, reducing manual effort. Static analysis is helpful to quickly identify potential threats, reducing the time and effort needed to detect them, which could take several hours or even days without automation.

Modern static analysis tools run continuously on the code base and new changes, similar to continuous integration (CI). Every time a team member makes a new commit, the changes are analyzed to detect the issues. That shifts the process of finding and fixing code health issues left.



Benefits of Continuous Quality

Continuous Quality (CQ) is a software development practice where code changes are immediately evaluated for their impact on the code's quality and maintainability and reported before being incorporated into the codebase. The objective of CQ is to provide quick feedback, enabling the identification and resolution of issues that may harm the maintainability of the code or escalate technical debt.

Maintaining code health is vital, and practices like peer-review of code, static analysis checks, and tracking key metrics like documentation coverage, test coverage, etc., help in this endeavor. Implementing CQ is a formal way to combine all these practices in the software development workflow. When combined with practices like CI and CD, CQ helps ensure that the team can deliver reliable software faster.

Some key benefits of implementing Continuous Quality as part of the development workflow include:

MORE RELIABLE AND SECURE SOFTWARE

A fundamental principle of CQ is to identify defects in code, such as anti-patterns, bug risks, and security vulnerabilities, as early as possible in the development process. By bringing these issues to the developer's attention while they can still be easily addressed, the likelihood of fixing them is increased compared to when they are discovered later. By repeating this process, frequently occurring problems have a higher chance of never making it into the codebase, resulting in more reliable and secure software free of defects.

FASTER TIME-TO-MARKET

CQ automates a significant portion of the code review process typically carried out by experienced developers. Identifying and fixing even minor issues before the review are typically resolved before it takes place, saving junior and senior developers time. That allows them to focus on more important tasks, resulting in a more streamlined and efficient code review process.

The improved reliability provided by CQ also minimizes the need for manual testing and verification, speeding up releases in production.

REDUCED COST OF SOFTWARE MAINTENANCE

Maintenance tasks such as debugging, refactoring, fixing broken dependencies, adapting code to new requirements, etc., consume a significant amount of developer time and focus. Insufficient documentation and a lack of quality processes can further complicate these tasks. CQ ensures that basic code hygiene and source code health metrics are maintained, making adding new modules, expanding existing functionality, or porting software to a new environment more manageable.

BETTER ESTIMATION OF RELEASE TIMELINES

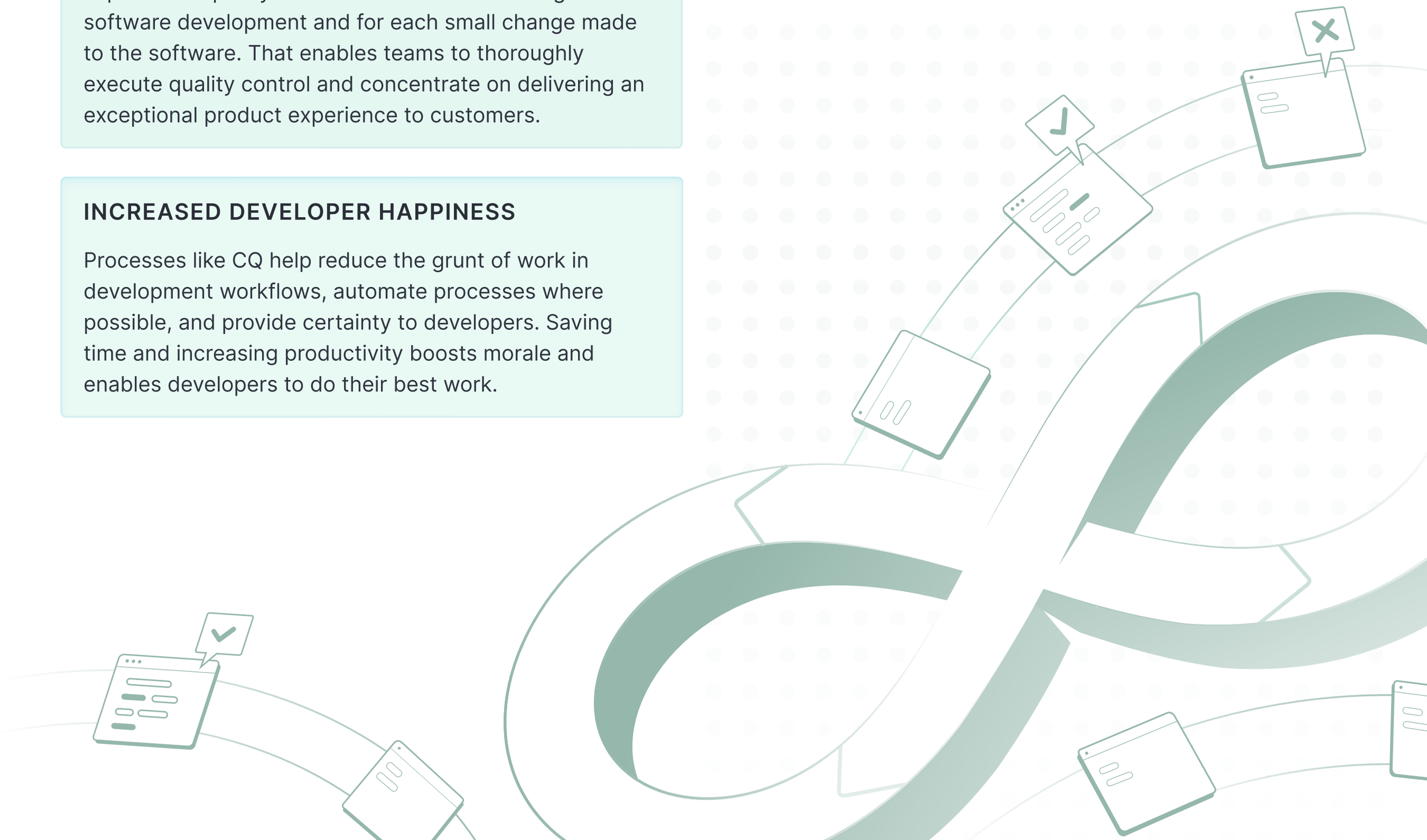
Accurately estimating the time required to deliver a new feature is crucial for determining the go-to-market strategy and associated revenue streams. Doing so without understanding the codebase's current state can lead to inaccurate predictions. CQ provides decision-makers with the necessary information to make more informed and realistic estimates of the release timeline.

IMPROVED CUSTOMER SATISFACTION

Customers desire a stable and secure product that functions as advertised without crashing. Building software can be challenging, especially at scale. CQ implements quality control from the earliest stages of software development and for each small change made to the software. That enables teams to thoroughly execute quality control and concentrate on delivering an exceptional product experience to customers.

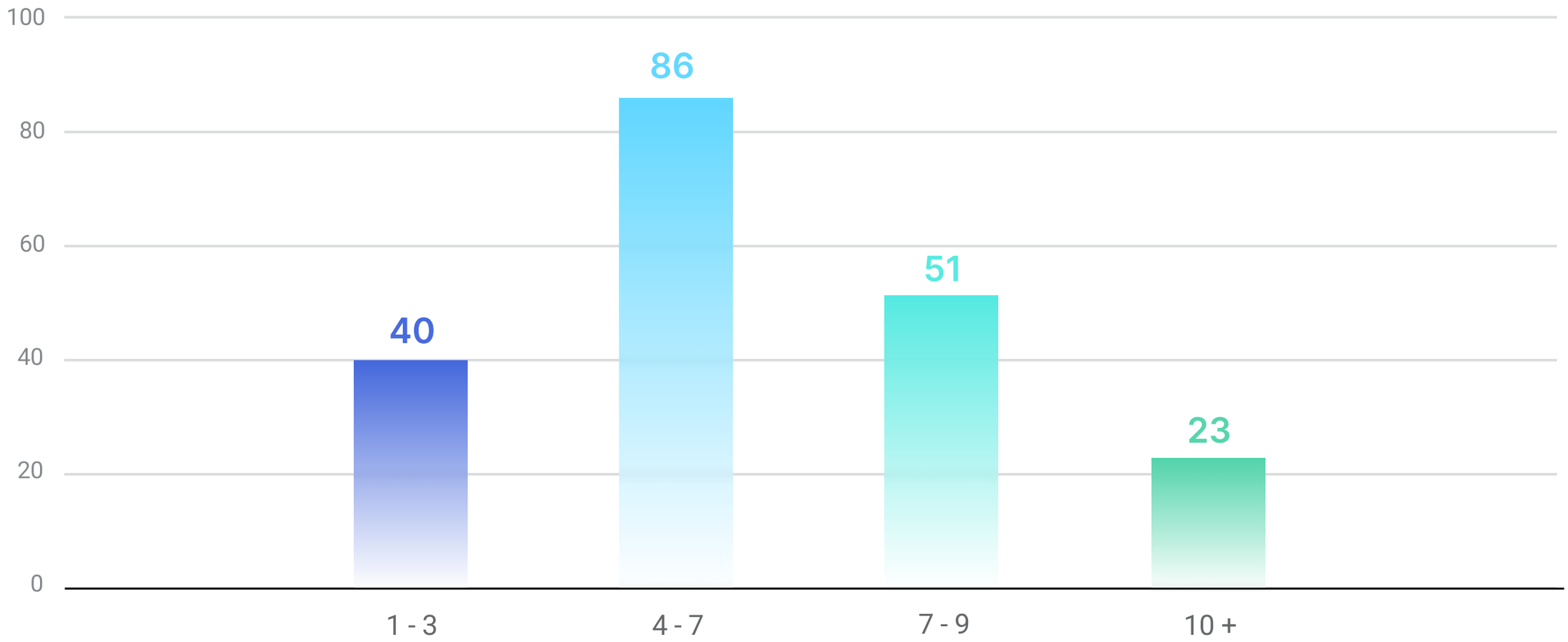
INCREASED DEVELOPER HAPPINESS

Processes like CQ help reduce the grunt of work in development workflows, automate processes where possible, and provide certainty to developers. Saving time and increasing productivity boosts morale and enables developers to do their best work.



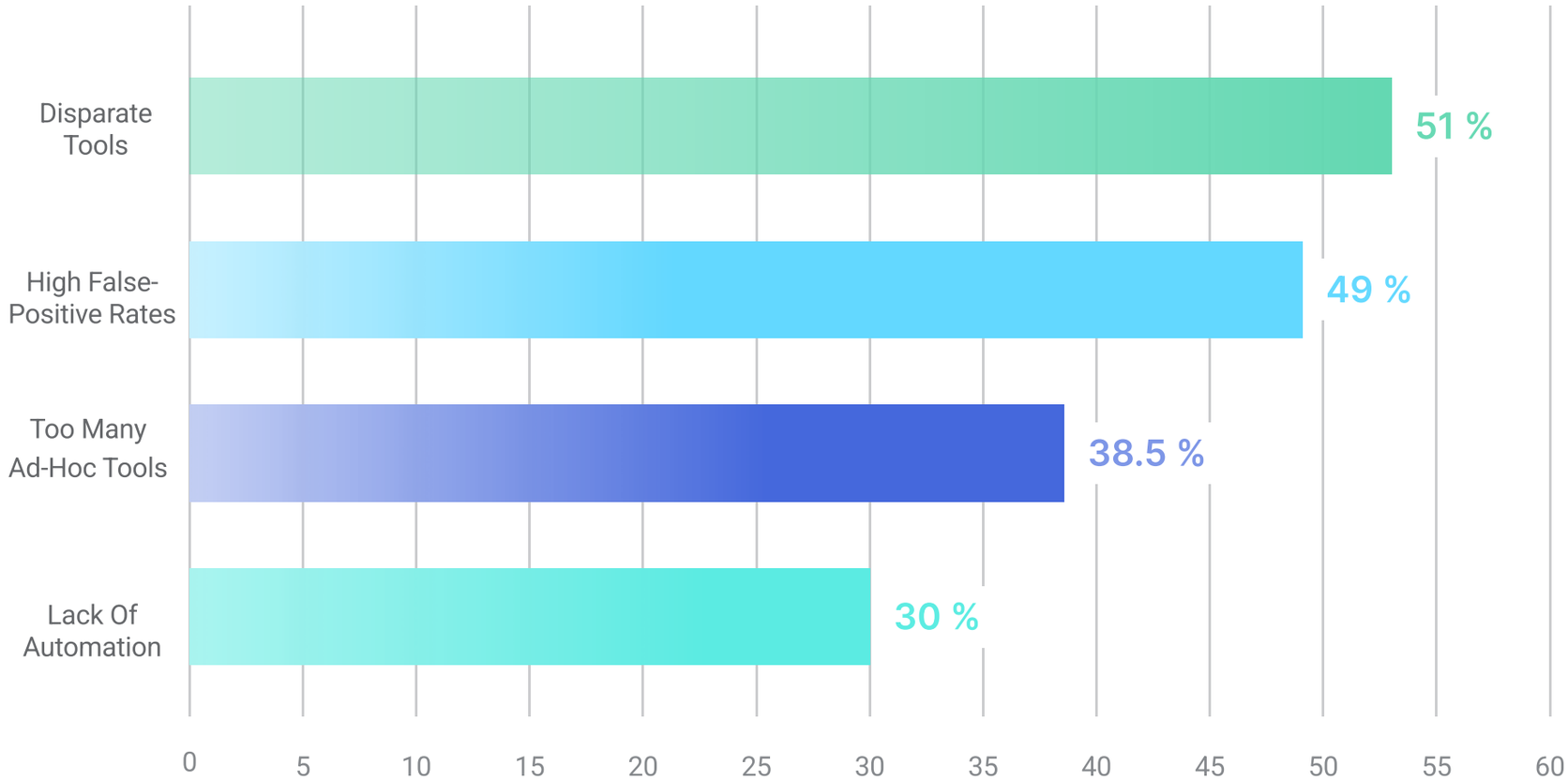
Up to 80% of respondents currently use at least four different tools to ensure the quality and security of their code. Thirty-seven percent of respondents use at least seven tools to achieve the same end.

How many tools do you use to track and improve the quality of your code?



51% percent of respondents selected “disparate tools” as one of the issues associated with their current code analysis process.

Some of the issues with my current code quality and security assurance process include (select all that apply)



Linting tools help automate code reviews. They perform basic static code analysis by flagging programming errors, bugs, style issues, and security vulnerabilities before the code is compiled and runs. While integrating these tools into your workflow is largely beneficial, some drawbacks may make them the wrong choice for long-term or more sophisticated software development.

Downsides of Too Many Code Health Tools

INCREASED COMPLEXITY AND COST

Using multiple tools can make the code review process more complex and time-consuming. Integrating all the tools into a unified workflow can be challenging, leading to a lack of consistency and increased effort. Each tool also comes with its price and license fees. Relying on too many tools can significantly increase the cost of ensuring code quality.

DECREASED COLLABORATION

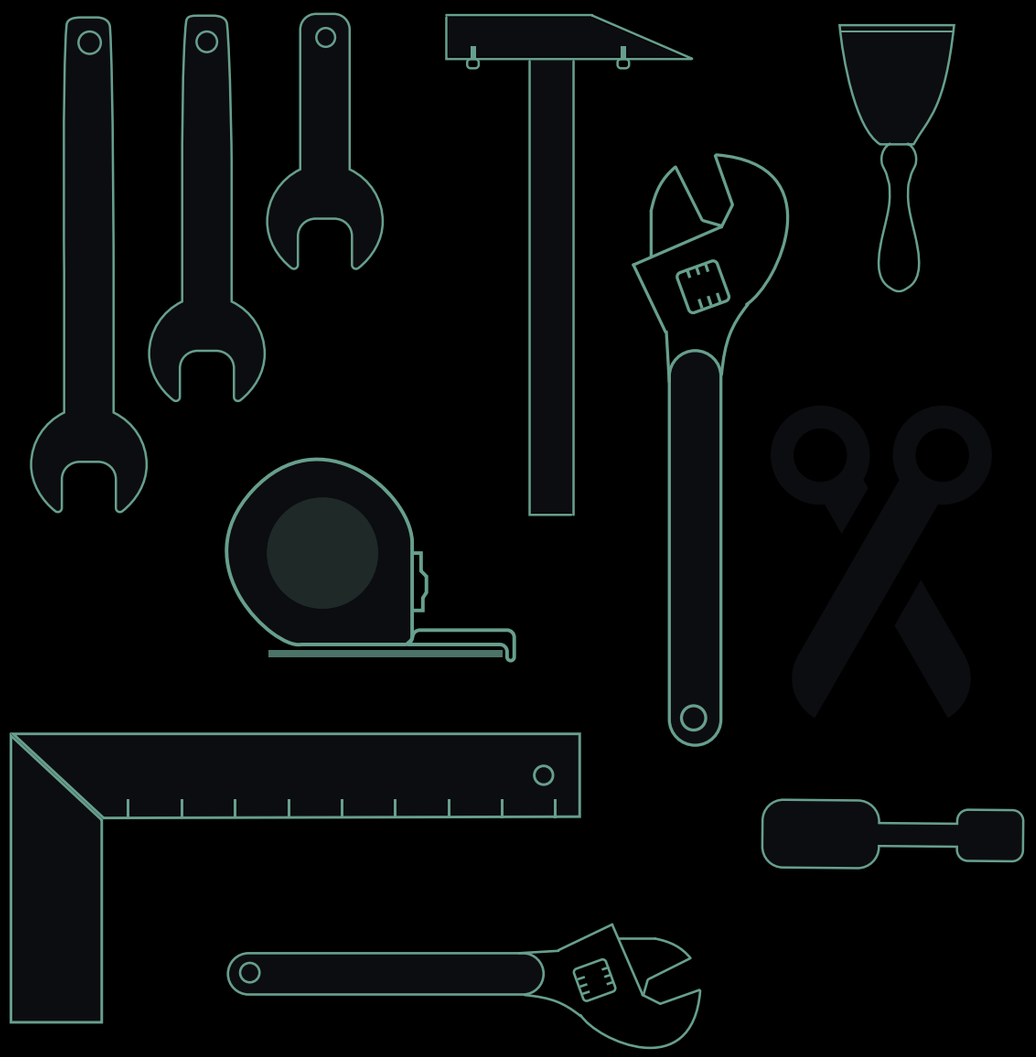
Using multiple tools can make it harder for teams to collaborate effectively, as each tool may have different ways of communicating and sharing information.

MAINTENANCE CHALLENGES

Maintaining and updating multiple tools can be challenging and time-consuming. Keeping up with updates and changes to each tool can quickly become overwhelming.

DIFFERING FUNCTIONALITY SETS ACROSS CODE ANALYSIS TOOLS

The lack of standardization among tools with different functions creates difficulties. Some tools concentrate solely on a single aspect of code quality, such as security, syntax, code coverage, or code style. Multiple tools perform various functions, making it challenging to view code health comprehensively. Finally, having the same errors highlighted by numerous tools can be irritating.



Benefits of Streamlining Code Health Tooling

BETTER ERROR DETECTION

By reducing the number of tools, developers can be more confident in the accuracy of error detection, as there is less chance of encountering conflicting information coming from different tools. That can help reduce confusion and increase productivity.

INCREASED EFFICIENCY

By reducing the number of tools, the development process becomes less complex, leading to a more efficient and effective workflow.

ENHANCED COLLABORATION

Using fewer tools reduces the risk of confusion over who discovered and resolved issues, leading to improved communication and collaboration between developers. That can result in a more cohesive and effective development team.

COST SAVINGS

Organizations can achieve cost savings by decreasing the number of tools they use, as they would not need to invest in and maintain numerous separate tools. That can lead to significant savings, particularly for organizations with limited resources

Challenges with High False-Positive Rates

Almost half of the surveyed selected “high false-positive rates” as one of the issues with their current code quality process. When quality tools alert developers with false positives too often, developers start dismissing them. A potential habit of treating everything as a false positive follows, eventually erasing the benefits of static code analysis.

LONGER DIFFERENTIATION TIME

Developers often have to manually distinguish real issues from false positives, resulting in extra time and effort to identify and resolve the issues discovered. That requires manual and automated solutions, making the process more time-consuming and inefficient.

LACK OF CONFIDENCE IN CODE REPORTS

The increased presence of false positives in code analysis reports can lead developers to disregard information they disagree with. That can cause them to focus on what they think could be wrong instead of what is wrong, leading to a greater chance of overlooking severe issues. Those more confident in their coding may likely ignore such problems.

CONFUSION OVER STANDARDIZATION

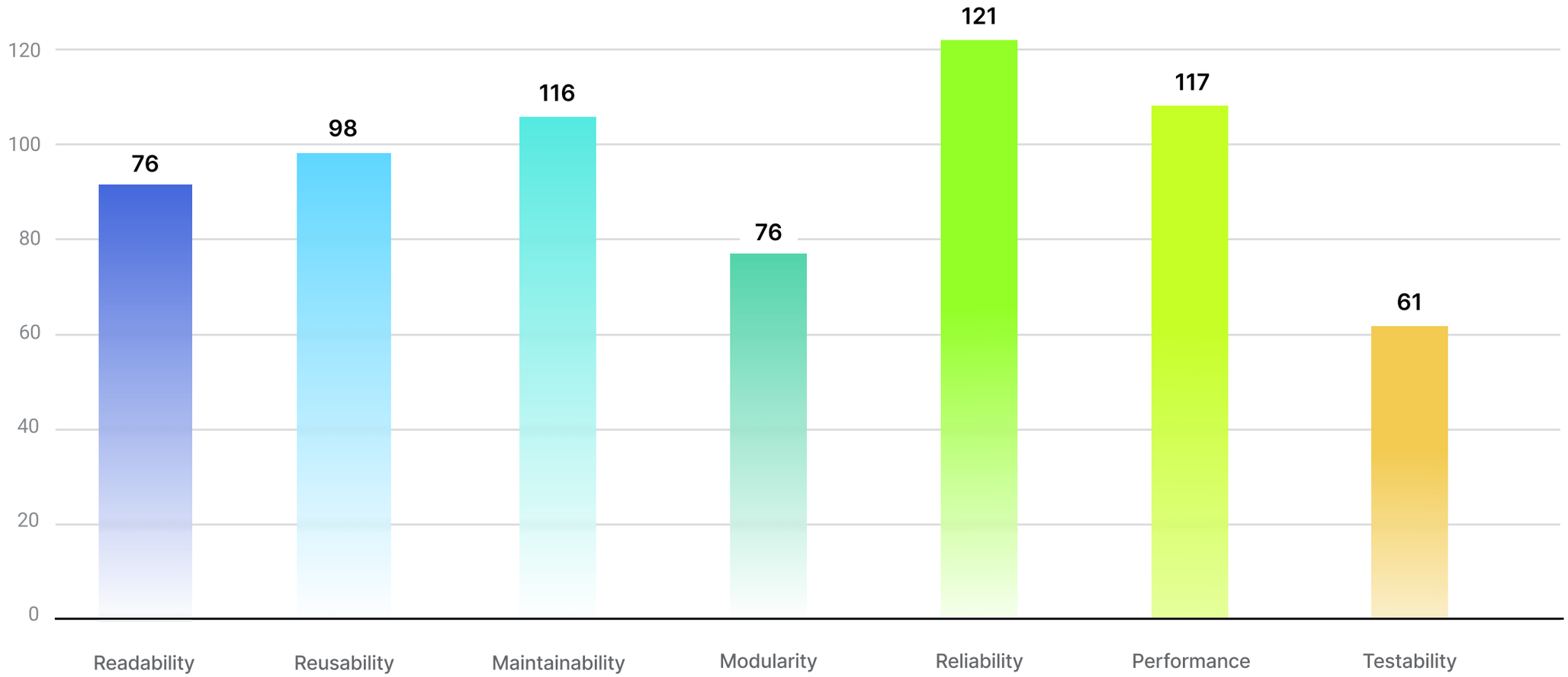
As developers discuss what constitutes a genuine problem never ends, creating definite regulations for code quality becomes increasingly challenging. Even if they agree on the flaws identified in one iteration, similar issues will likely resurface in future iterations, demonstrating no progress in producing better code from the onset.

Although developers may simply adjust the linter configuration to prevent false positives, doing so across several liters (with varying settings) is cumbersome and heightens the risk of human error. Some tools have incomplete rules, and there might be inconsistencies in setting updates, which makes the argument for streamlining the code quality process even more compelling.

Reliability, Performance, and Maintainability

According to our survey, the respondents ranked "Reliability" as the most critical aspect to consider when assessing the health of their codebase, followed by "Maintainability" and "Performance." These three aspects are essential for a codebase's overall health and effectiveness.

What do you consider when measuring the quality of your code (check all that apply)?



Reliability is a crucial aspect of a healthy codebase, as it is vital to building and maintaining the trust of users and customers. A reliable codebase functions as intended, even under heavy use and stress, which helps to ensure user satisfaction and confidence in the application.

Maintainability plays a role in reliability as well. It refers to the ease of making changes and updates to the software, which can significantly impact the overall development cost. There are reports that software maintenance costs can range from 40% to over 90% of the total cost of development[6].

Teams can improve maintainability by avoiding complexity and code duplication, ensuring readability, and proper documentation. Here, code coverage helps to identify any areas of the code that are not being tested and are, therefore, more likely to contain bugs or be challenging to maintain.

Performance is about how well the code does what it's supposed to do. It involves comparing how fast the code can be executed and how many resources it requires to function as loads rise. Performance points are a great way to measure the overall quality of a software system. They can be an effective tool for judging the success of a software project.

[6] "App Maintenance Cost Can Be Three Times Higher than ... - Techstep." 2 Nov. 2021 https://www.techstep.io/articles/app-maintenance-cost-can-be-three-times-higher-than-development-cost. Accessed 1 Feb. 2023.

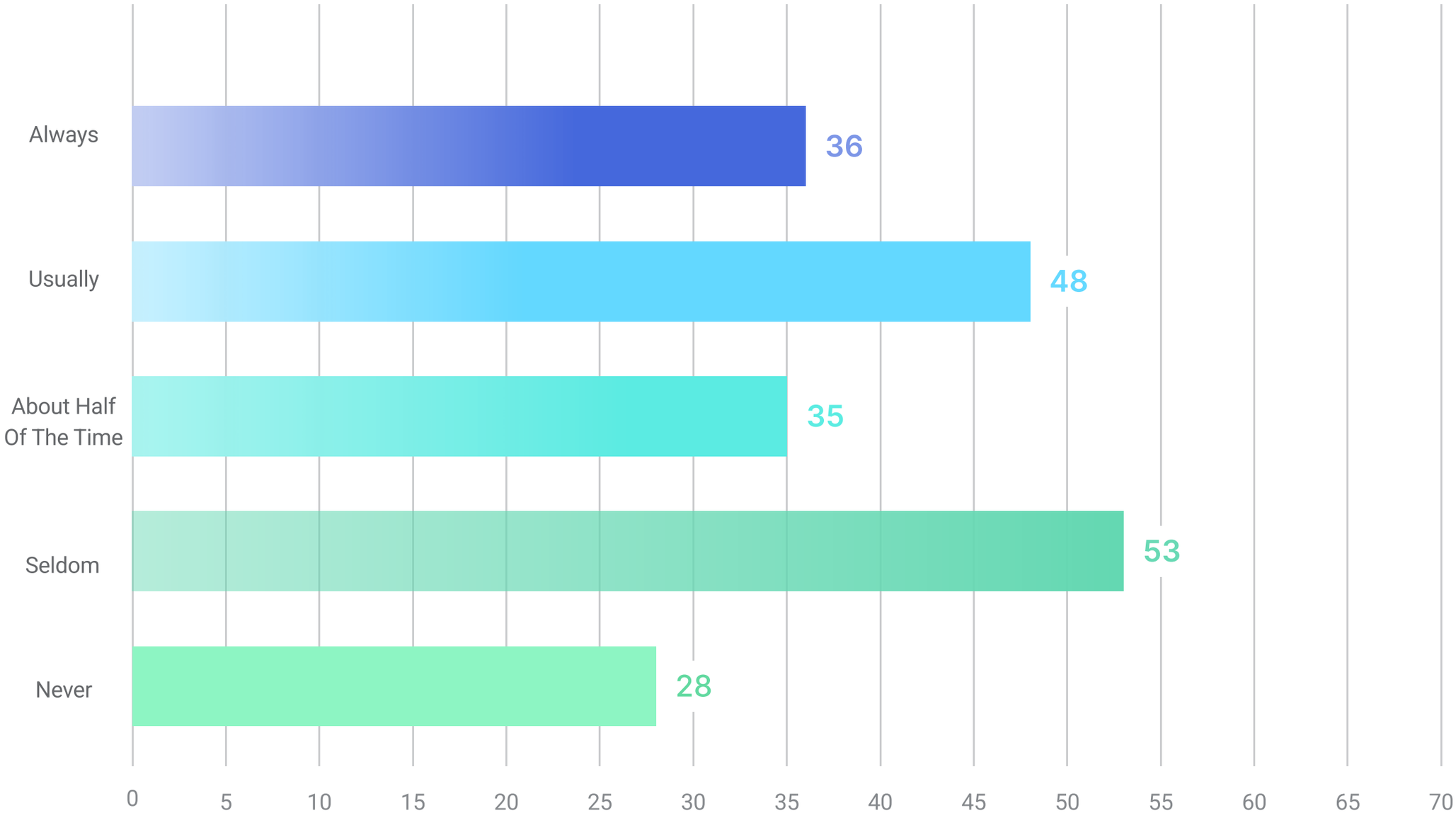
SECTION TWO

Code Health: Outcomes

Impact on Developer Productivity

Nearly 60% of the respondents state that their code health screening process caused delays at least half the time.

Ensuring the quality and security of my codebase has caused me or my team to miss delivery deadlines

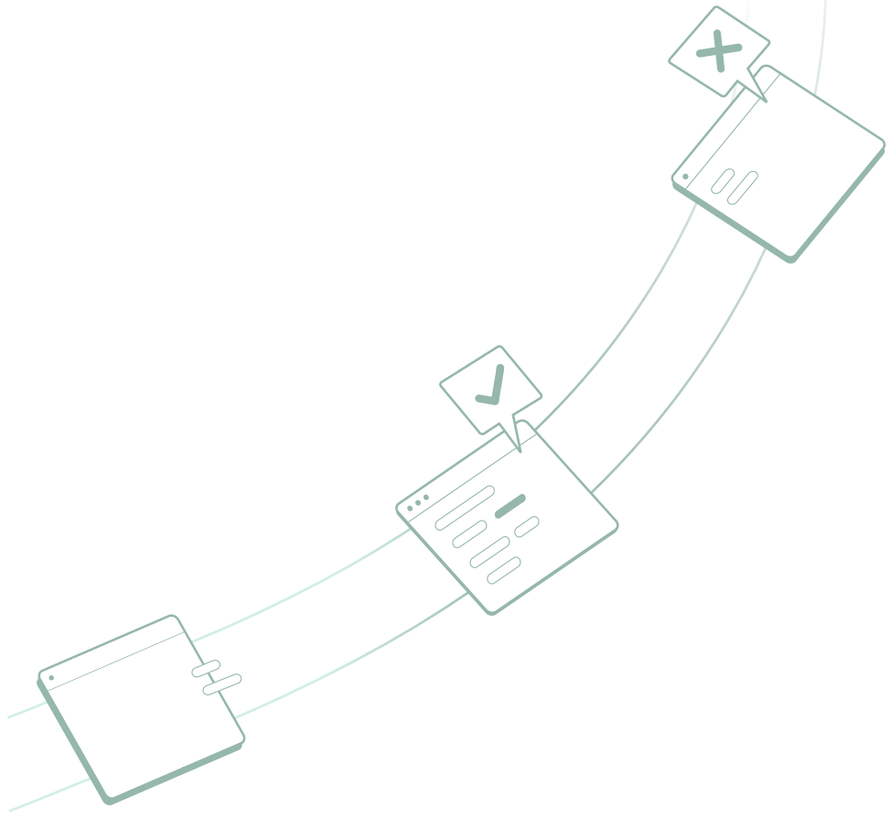


Ensuring the overall quality of the code to avoid technical debt should not keep dev teams from meeting deadlines. Shifting quality and security left is now the standard. By catching issues earlier, fixing them is easier and more affordable and helps prevent issues from being deployed to production. This approach also helps improve collaboration between development, security, and operations teams, as they work together to ensure the quality and security of the code.

Additionally, shifting security and quality left helps meet the increasing demand for faster and more frequent software releases while maintaining the highest quality and security standards. Shipping quality code without compromising speed is the ultimate differentiator, as it allows businesses to offer new products and features at a competitive pace.

Solutions that help optimize code deliveries should also provide visibility into the development process, allowing software and security leaders to stay aligned with other stakeholders in the organization via reports. This visibility ensures that everyone is aware of the progress toward business goals and any potential risks or issues that may arise. It can also help leaders make data-driven decisions and allocate resources more effectively.

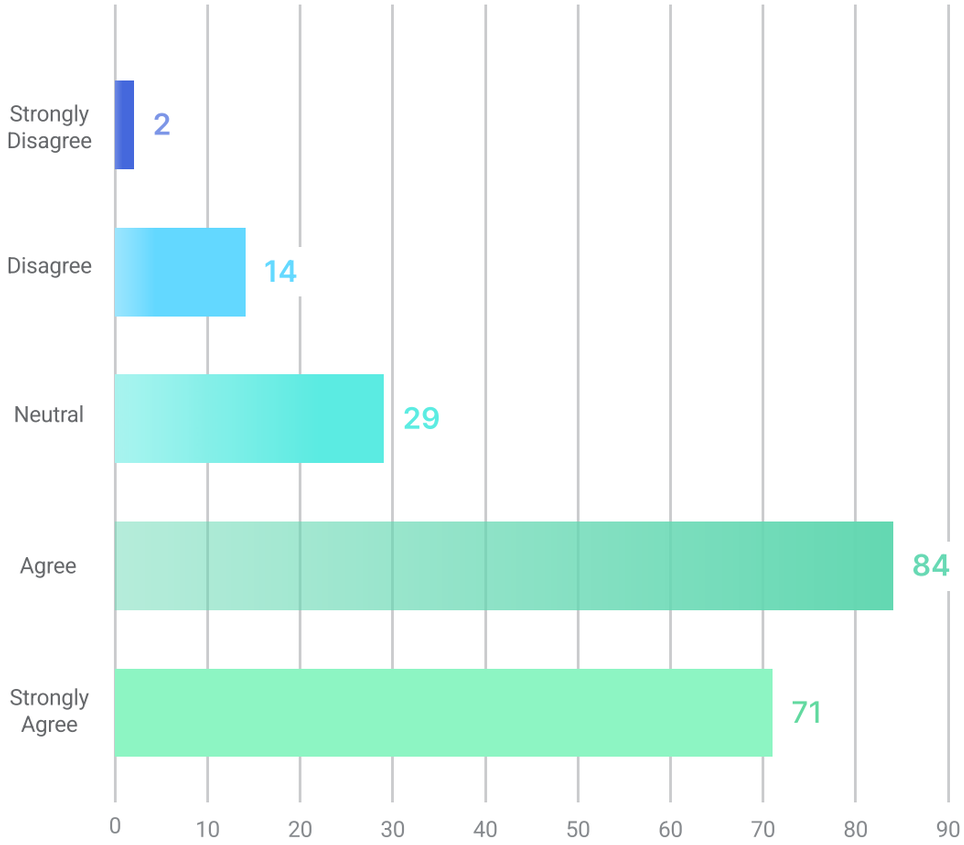
Reports that provide insights into the health of the code can help technical leaders and business stakeholders stay aligned by offering a shared understanding of the software's security posture, performance, and overall quality. Such reports can provide a clear picture of any vulnerabilities and risks associated with the software, enabling the team to prioritize issues and allocate resources effectively. Those reports can help stakeholders make informed decisions on budget, timelines, and other factors, allowing them to align business goals with technical needs.



Impact on Onboarding

A silver lining: approximately 80% of the respondents agree that their code review process has helped their teams improve developer onboarding and expand their skill set.

> **Our code analysis process has facilitated developer onboarding**



In the United States alone, the number of people working from home tripled between 2019 and 2021, according to the United States Census Bureau[7]. Dev leaders should use solutions to identify and address skills gaps without over-the-shoulder reviews and face-to-face interactions.

More than fixing current code issues is required to prevent them from recurring before the next release. It's essential to also focus on developing the skills of team members to ensure sustainable improvements and prevent future issues. Static Analysis can help close the knowledge gap in software development teams by providing a comprehensive view of the code, identifying areas for improvement, and providing actionable insights for further development.

Providing real-time visibility of code issues and how to fix them through ongoing professional development is an essential aspect of efficient deliveries. This approach empowers developers with the knowledge and skills to identify and address problems as they arise rather than waiting for them to be discovered later in the development process. That can lead to faster delivery times, improved code quality, and a more confident and skilled development team.

Offering developers the tools and resources they need to improve their skills continuously helps to create a culture of continuous learning and development, which can further enhance the overall efficiency of the software development process..

[7] "The Number of People Primarily Working From Home Tripled ...." 15 Sep. 2022, https://www.census.gov/newsroom/press-releases/2022/people-working-from-home.html. Accessed 1 Feb. 2023.

Conclusion

Developers can elevate their confidence and efficiency by using streamlined tooling, effective automation, and comprehensive reporting to ensure code health. They can also promote secure and stable software, reduce the cost of developing and delivering software, speed up producing high-quality products and features, and improve your brand's reputation.

About Us

DeepSource is the code health platform, providing software developers with the resources they need to create maintainable and secure code, leading to increased software stability and faster development times.

Schedule a demo today, or visit our website to learn more about us.

Schedule a Demo

Trusted by 3,700 companies, from startups to Fortune 500s

twilio

privacy dynamics

Qubole

heycar

appsmith_

iterative