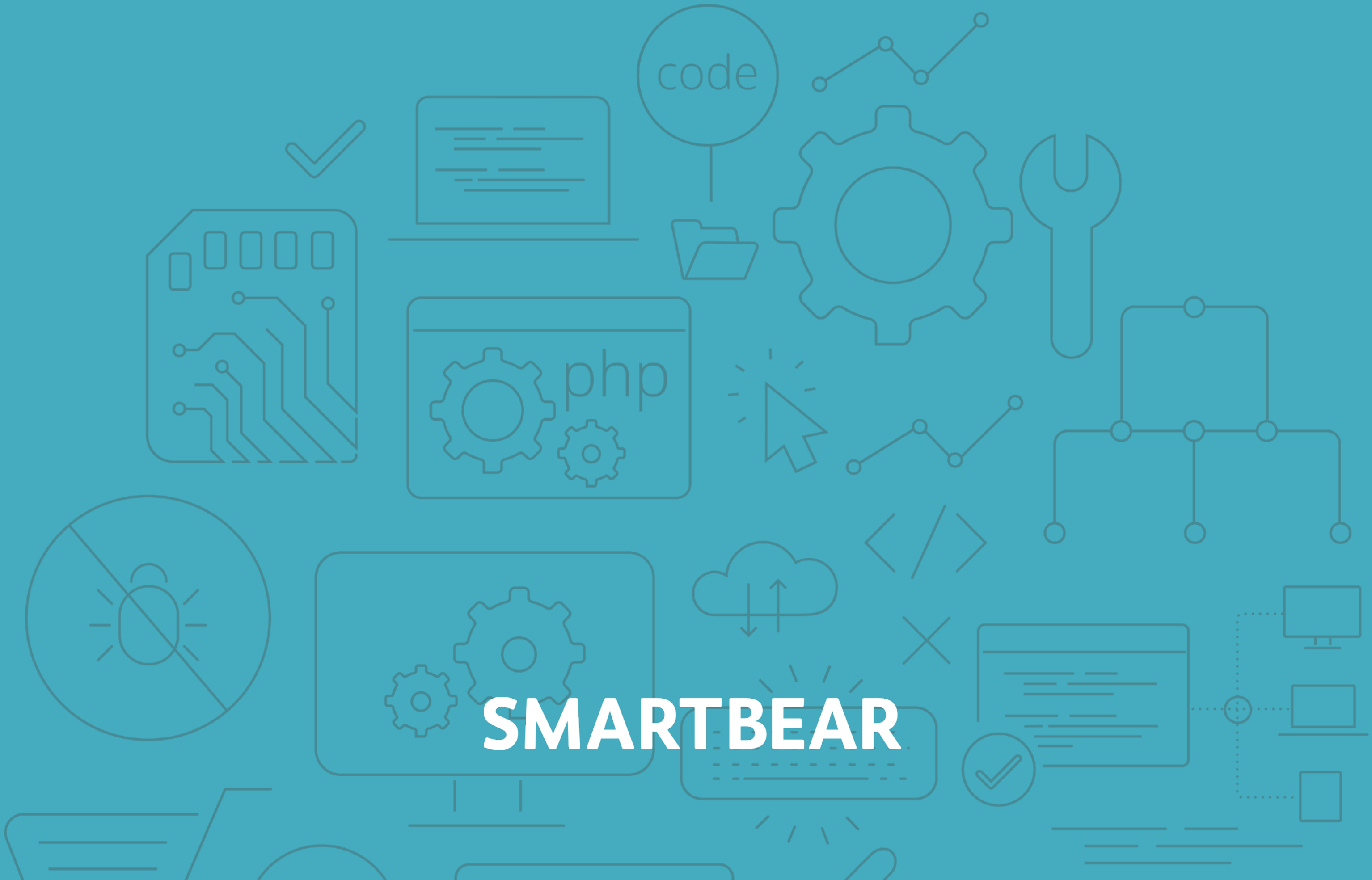
Collaborator

The State of Code Review 2017:

Trends & Insights into Dev Collaboration



SMARTBEAR

Table of Contents

Preface:

This survey was designed to establish benchmarks for the software industry concerning the ways in which organizations are developing and maintaining the quality of software in 2017. The structure of this year's report closely mirrors the 2016 report structure so that we may provide meaningful year-over-year comparisons and comment on significant trends around code quality. This report covers the following topics:

- Perceptions Surrounding Code Quality.
- Approaches to Code Review.
- Code Review Tools and Decision Making.
- Software Development Tools Methodology.

Methodology:

To ensure the 2017 report data is relevant SmartBear Software conducted a global online survey over the course of four weeks during the months of December 2016 and January 2017. The findings presented are based upon the aggregated responses from more than 550 software developers, testers, IT/operations professionals, and business leaders representing more than 30 different industries. Participants in the survey work at companies of all sizes, from fewer than 25 employees to over 10,000. Similarly, their software teams range in size from fewer than 5 to more than 50 team members.

Key Findings

Two-thirds of respondents are satisfied with the quality of software they help build.

- Nearly two-thirds of respondents said they either agree (52.4%) or strongly agree (12.3%) that they are satisfied with the quality of the software they deliver.
- 13% of respondents either disagree (11.2%) or strongly disagree (1.8%) that they are satisfied with their software quality.
- The percentage of respondents citing code review as the best way to improve code quality **nearly doubled** between 2016 and 2017, jumping from 27.4% to 51.9%.

Code review continues to be the ***#1 way*** *to improve code quality*

When it comes to code quality, code review maintains the leadership position.

- Code review is still viewed as the #1 way to improve code quality.
- Unit testing, cited by 51.2% of respondents, also saw a sharp rise from year to year.
- 92% of respondents say that improved code quality is the biggest benefit of code review.
- 87% of respondents say that improving code quality is the biggest business driver determining the need for a code quality tool.

Nearly three-quarters of respondents are doing code review using a combination of methods, including ad-hoc processes, meetings, and tool-based reviews.

- 74.1% of respondents are doing code review.
- 74.6% are doing ad-hoc, or “over the shoulder” code review.
- 60.4% are doing tool-based code review.
- 56.8% are doing meeting-based code review.

Workload, time constraints, and lack of manpower are the biggest obstacles to code review.

- 55.1% of respondents say workload is the biggest obstacle preventing them from doing the level of code review they desire.
- 44.1% say deadlines/time constraints are one of the primary obstacles.
- 33.7% say that lack of manpower is one of the primary obstacles.

Using a tool for code review enables teams to review code on a more frequent basis.

- 60.4% of respondents are using at least one tool for code review.
- 21.4% of respondents who are doing tool-based code review are doing it daily; 27.1% are doing it on a weekly basis.
- 5.5% of respondents who are doing meeting-based code review are doing it daily; 23.7% are doing it on a weekly basis.
- 16.6% of respondents that are doing ad-hoc code review are doing it daily; 36.4% are doing it on a weekly basis.

Several tool categories are dominated by a single product.

- 57.4% of respondents are using JIRA for bug tracking.
- 57.3% of respondents are using Git for their SCM.
- 50.2% of respondents are using JIRA for requirements management.

Git and Subversion are the most commonly used software configuration management tools (SCMs).

- 57.3% of respondents are using Git as their SCM.
- 30.8% of respondents are using Subversion as their SCM.
- 27% of respondents are using GitHub or GitHub Enterprise for repository management.

Section 1: Code Quality in 2017

With software code driving competitive advantage in numerous products and processes across industries, organizations of all types and sizes view code quality as a top priority. But, are these organizations meeting the high code standards they set for themselves? How are they ensuring that they maintain the quality of their software? In the first section of the State of Code Review 2017 report, we look at how organizations are maintaining code quality in 2017.

Highlights

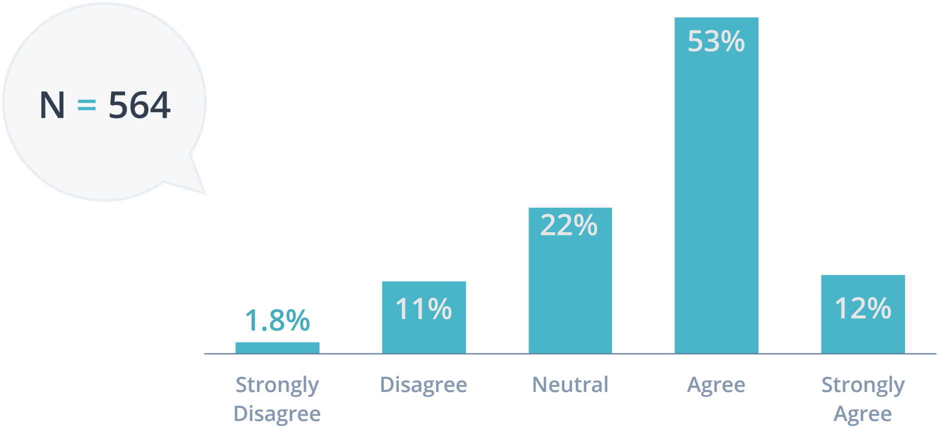
- Nearly two-thirds of respondents agree that they are satisfied with the quality of software they help build.
- Two-thirds of respondents say they are regularly able to get releases out on time.
- Code review is still viewed as the #1 way to improve code quality.

Two-thirds of respondents agree that they are satisfied with the quality of software they help build.

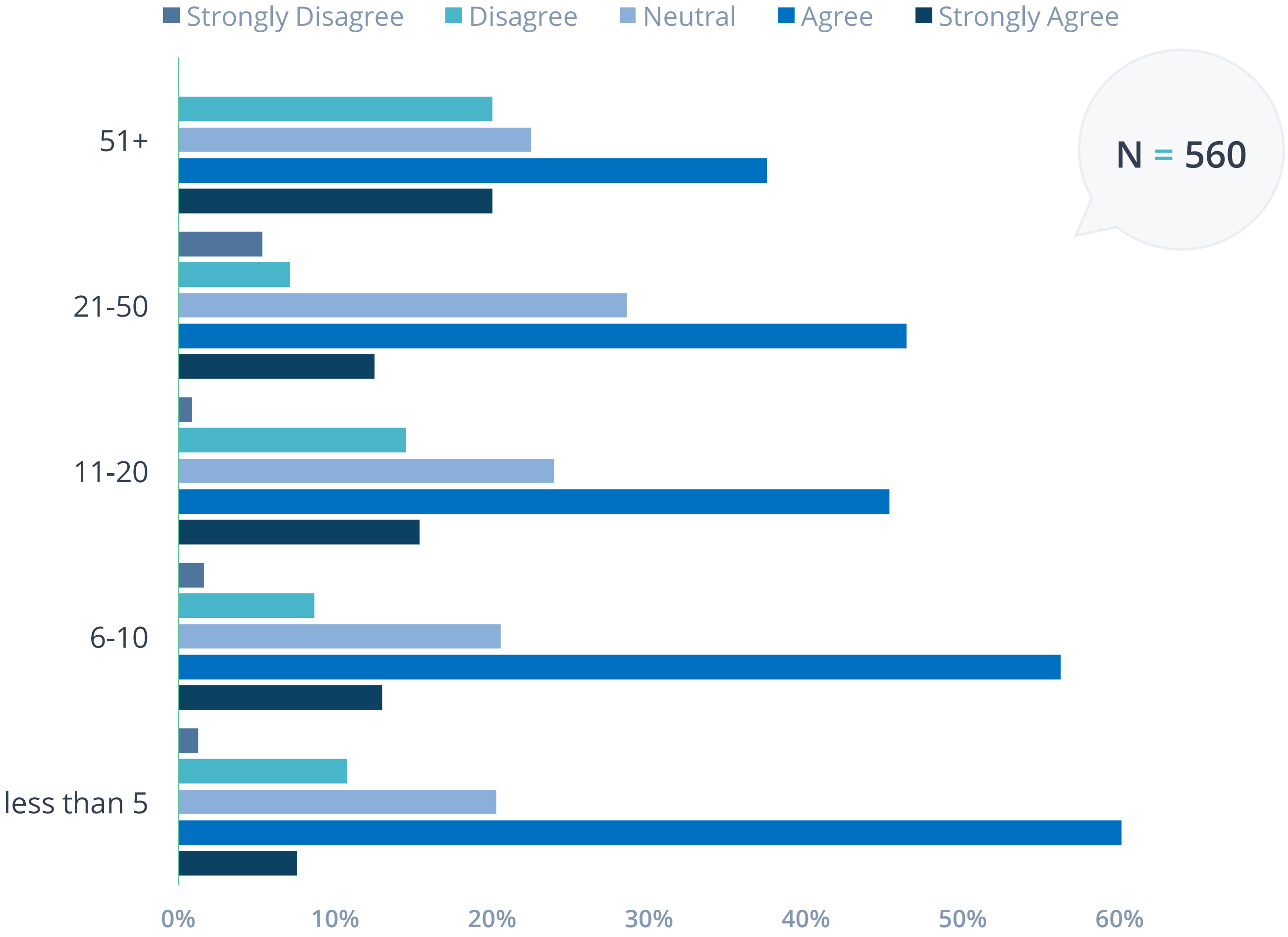
Overall, we found that respondents felt positively about the quality of the software they help build (64.9%). Of those who didn't either agree or strongly agree, just 13% said they felt negatively about the software they help develop. The 13% figure represents a 30% increase over the 10% who disagreed or strongly disagreed in 2016. Based on our observations across industries, we see several potential factors that may give rise to the higher rate of dissatisfaction. These include new regulations in certain industries, increasing pressure on teams to ship new code faster, or breakdowns in the development and QA processes.

The overall positive sentiment was felt across industries and roles within software teams. One area where negative sentiment was higher was within software teams of 50 people or more.

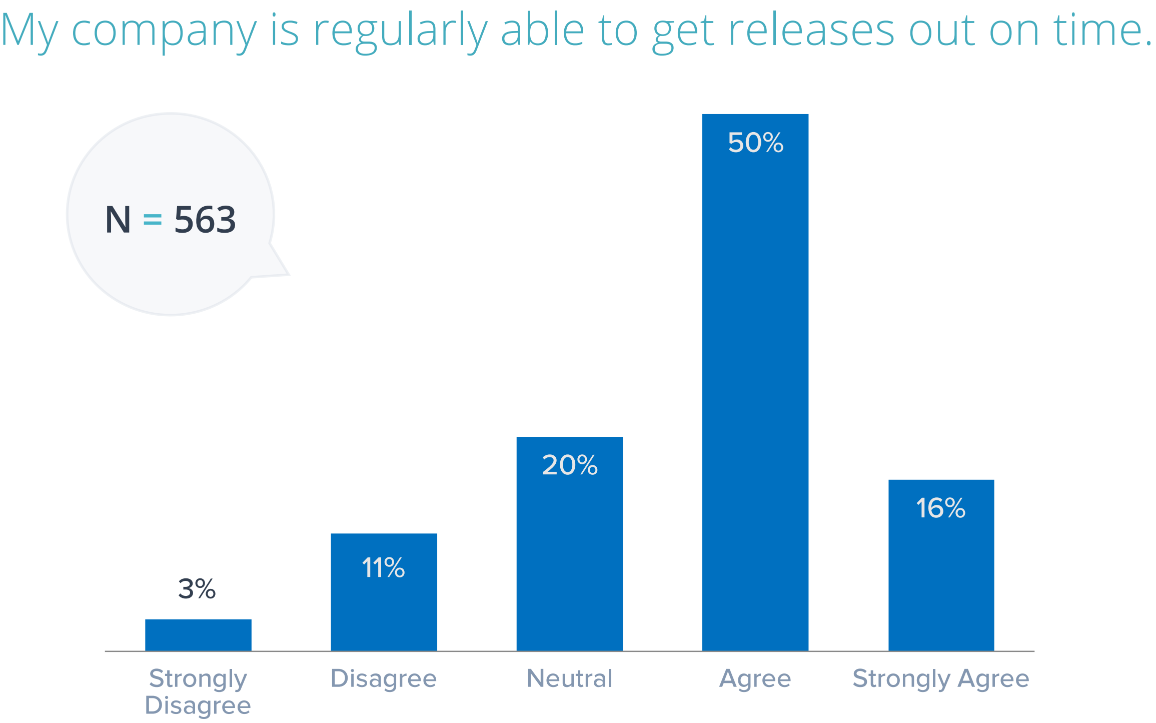
I am satisfied with the overall quality of the software I help deliver (specifically regarding performance, bugs, etc.)



1 in 5 respondents on development teams of 50+ said they either disagreed or strongly disagreed that they were satisfied with the quality of code they help build.



Two-thirds of respondents say they are regularly able to get releases out on time.



Development teams with 6-10 members were the most positive about their ability to get releases out on time, with 76.6% reporting that they either agree or strongly agree. That's a change from 2016, when teams with 21-50 members were the most positive (78.2%). That category dropped to just 55.4% in 2017 and drove the increasing rates of dissatisfaction.

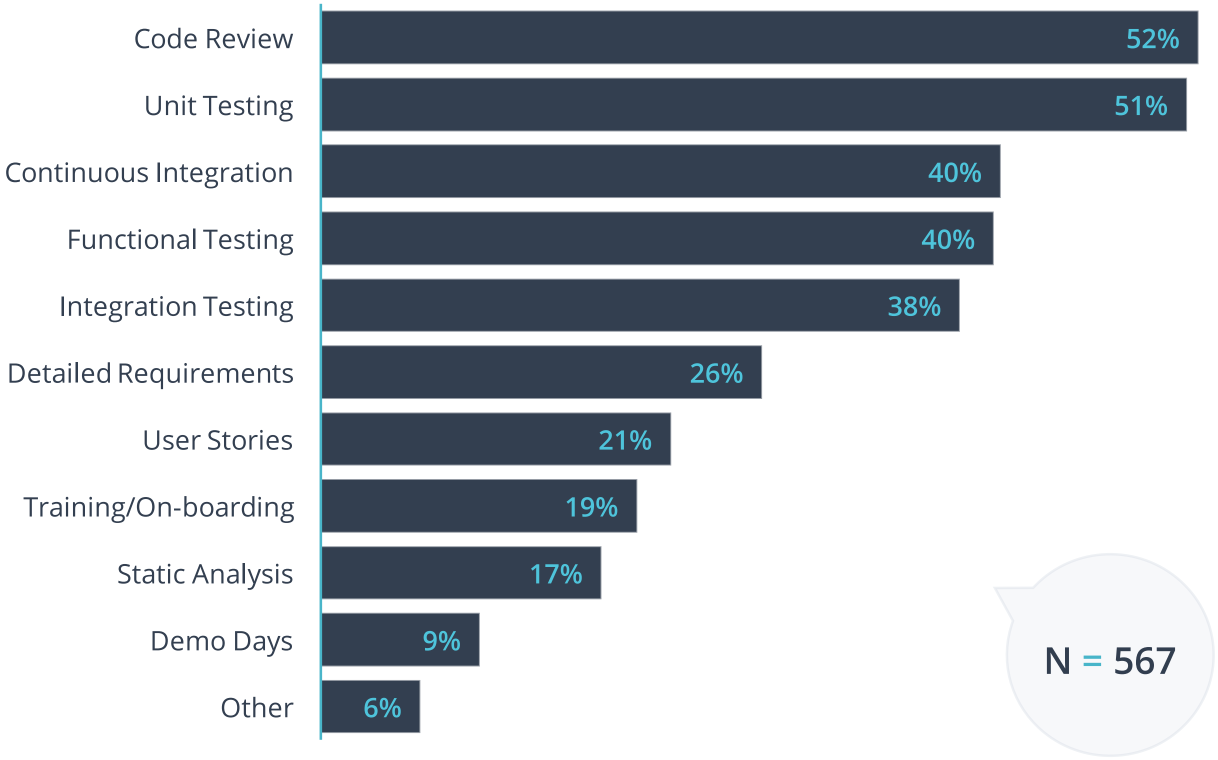
Code review continues to be viewed as the #1 way to improve code quality

When asked what the number one way to improve code quality is, 51.9% of respondents said that code review is the best way. That is nearly double the 27.4% of respondents favoring code review last year. We attribute most of the jump to increased spend on software quality. The growth in popularity of repository management solutions has provided more teams with basic tool based code review functionality.

Another dramatic increase occurred in relation to unit testing, which 51.2% of respondents cited as the best way to improve code quality in 2017, up from just 18.7% in

2016. This change reflects the overall shift in awareness and focus around quality which we saw increase greatly the last 12 months. Development teams and management recognized that code quality needed to improve. With the increased focus on continuous delivery and deployment, unit testing has become more critical due to condensed delivery time lines. The automated testing process allows teams to receive rapid feedback on the success or failure of changes before they hit production. Overall, testing is considered a critical method for improving code quality – 51.2% say unit testing, 39.8% say functional testing, and 37.8% say integration testing are primary ways to improve code quality.

What do you believe is the number one thing a company can do to improve code quality?



Section 2: Approaches to Code Review

Code review remains the number one method for improving code quality. Yet, different organizations – or different teams within organizations – vary in their methods for implementing code review. In the second section of the State of Code Review 2017 report, we look at how teams are using code review and the challenges that can often get in the way.

Highlights

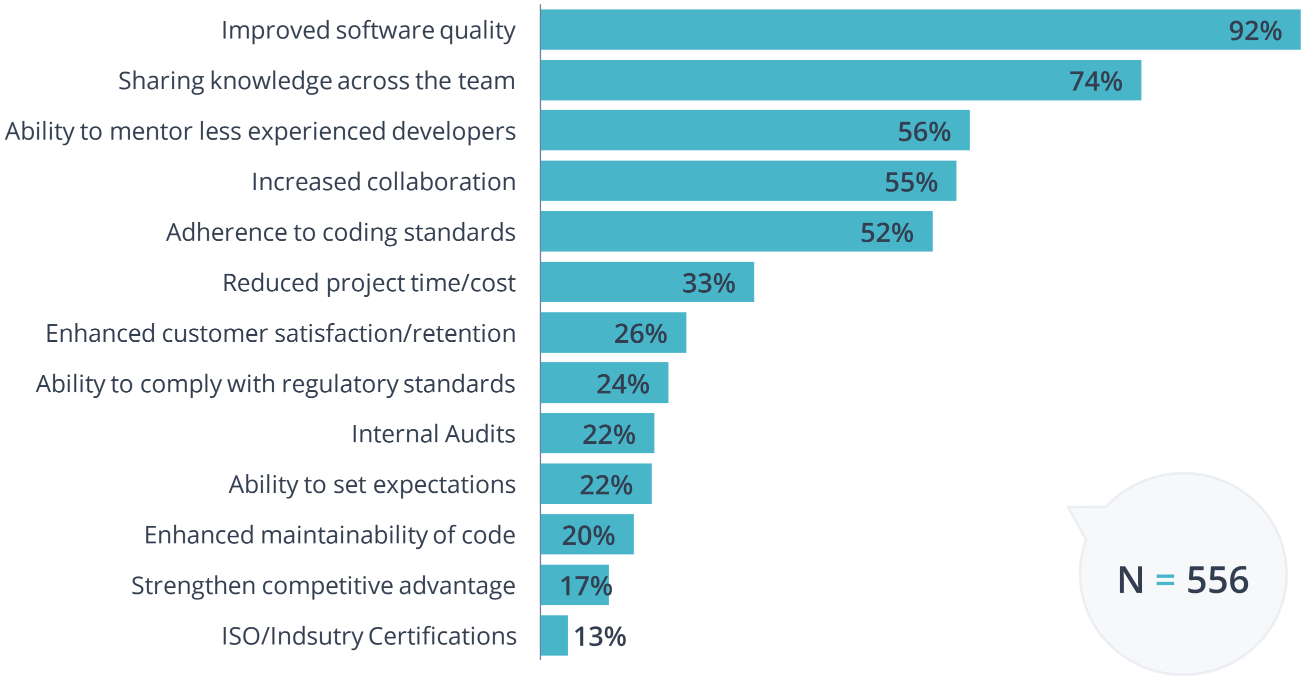
- 75% of respondents are doing "over the shoulder" code review; 68% are doing it at least once a month.
- 57% of respondents are doing meeting-based code review; 44% are doing it at least once a month.
- 60% of respondents are doing tool-based code review; 49% are doing it at least weekly.
- Workload (55.1%), time constraints (44.1%), and lack of manpower (33.7%) are the biggest obstacles to code review.

"Over the shoulder" code review and meeting-based code review climbed by 3 and 7 percentage points, respectively, over last year. Yet, for the most part, both are performed only monthly or less and lag far behind tool-based code review in daily or weekly reviews.

Ninety percent of respondents say that code quality is the best reason for doing code review.

Beyond the quality of code, reviews are also viewed as a primary method for improving the ways teams collaborate, including team-wide knowledge sharing, mentoring less experienced developers, and increasing the frequency of collaboration.

What do you believe are the most important benefits of code review?

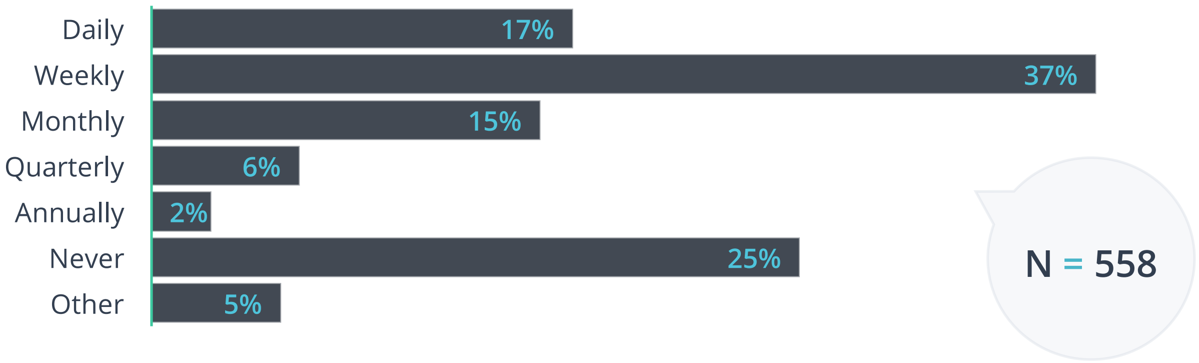


75% of respondents are doing “over the shoulder” code review; 68% are doing it at least once a month.

Ad-hoc, or “over the shoulder” code review is the most commonly used type of code review within organizations, with three-quarters of respondents participating in ad-hoc reviews throughout the year.

The majority of respondents who are doing ad-hoc code review do it at least once per month, with a majority of respondents conducting ad-hoc code review on a weekly basis.

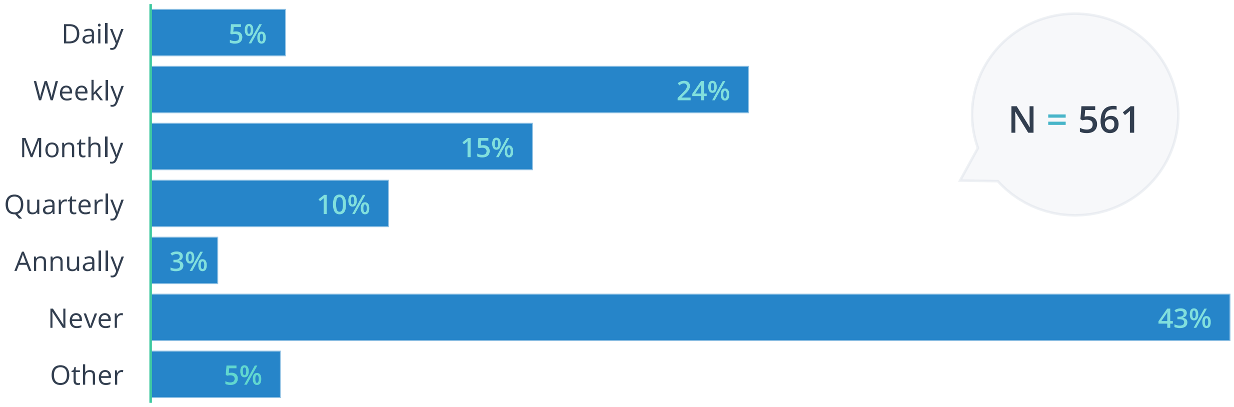
How often do you participate in an ad-hoc (“over the shoulder”) code-review process?



57% of respondents are doing meeting-based code review; 44% are doing it at least once a month.

Meeting-based code review is the least commonly used method for code review, with a little more than half of organizations participating in meeting-based reviews.

How often do you participate in a meeting-based code-review process?

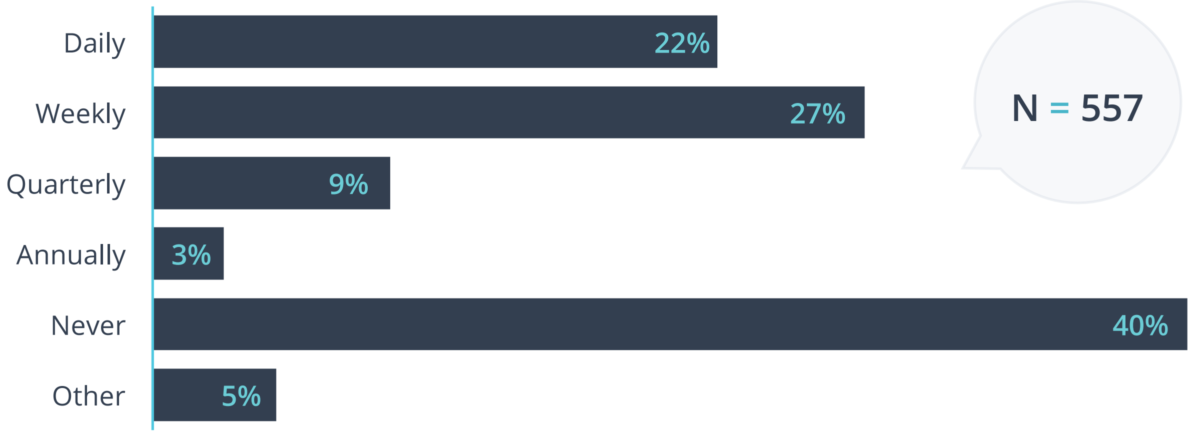


60% of respondents are doing tool-based code review; 49% are doing it at least weekly.

Six out of every ten individuals surveyed say that their organization are using a tool to help with code review. The 60% number represents a 3-point drop from last year, which we attribute to a larger number of this year's survey respondents coming from smaller development teams, specifically those with 10 or fewer members. Such teams find it easy and convenient to perform ad-hoc reviews, as they are often seated together in a single physical location. For many small teams, combining a collaboration tool (e.g. Slack) with ad-hoc reviews is sufficient.

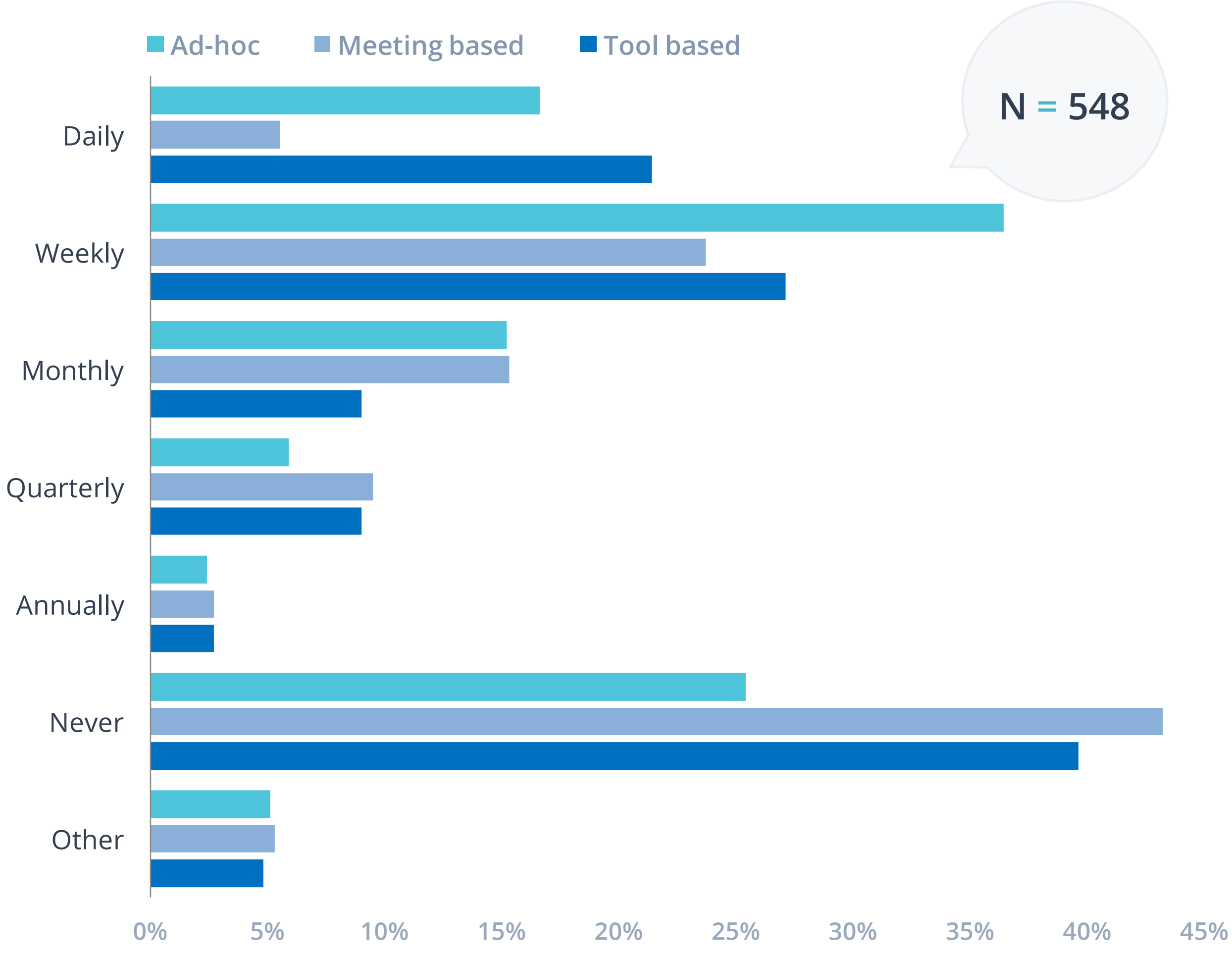
Nevertheless, organizations that are doing tool-based code review are doing it on a more consistent basis, with 49% doing it at least weekly versus 45% in last year's survey. It is also evident from the survey data that having a tool enables more frequent code review. Twenty-one percent of respondents perform tool-based code reviews daily, which is 30% higher (or more) when compared with other review methods.

How often do you participate in a tool based code review process?



21.4% of organizations are doing tool-based code review daily, compared to 16.6% for ad-hoc code review and just 5.5% for meeting-based code review.

How often do you participate in an ad-hoc, meeting-based, or tool-based code-review process?

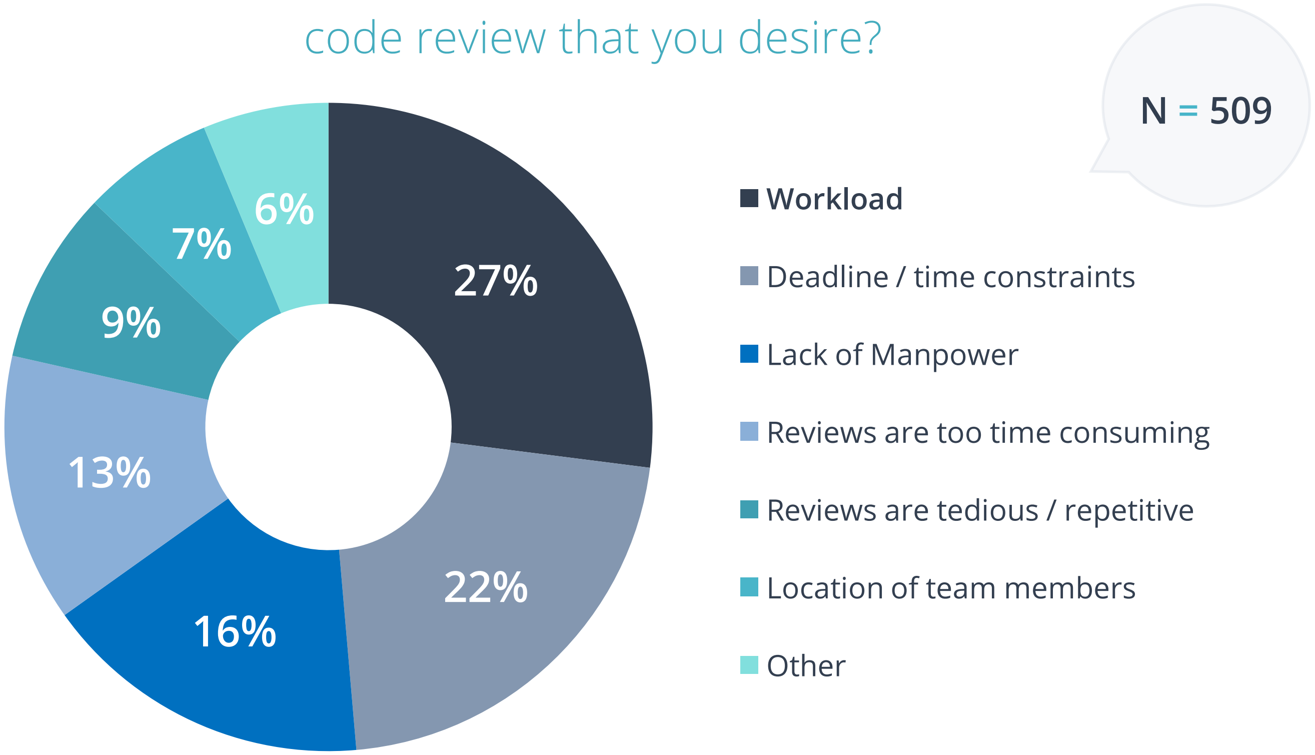


Workload, time constraints, and manpower are the biggest obstacles to code review.

While teams understand the importance of code review, they often face obstacles when it comes to implementing it into their team or organization. Workload is the number one obstacle facing their teams when it comes to doing code review. However, only 55% of respondents cited workload as an obstacle this year – a significant drop from last year's 66%. This is good news for development teams. It demonstrates how companies have stepped up hiring for their software development teams and are lowering the work burden placed on team members.

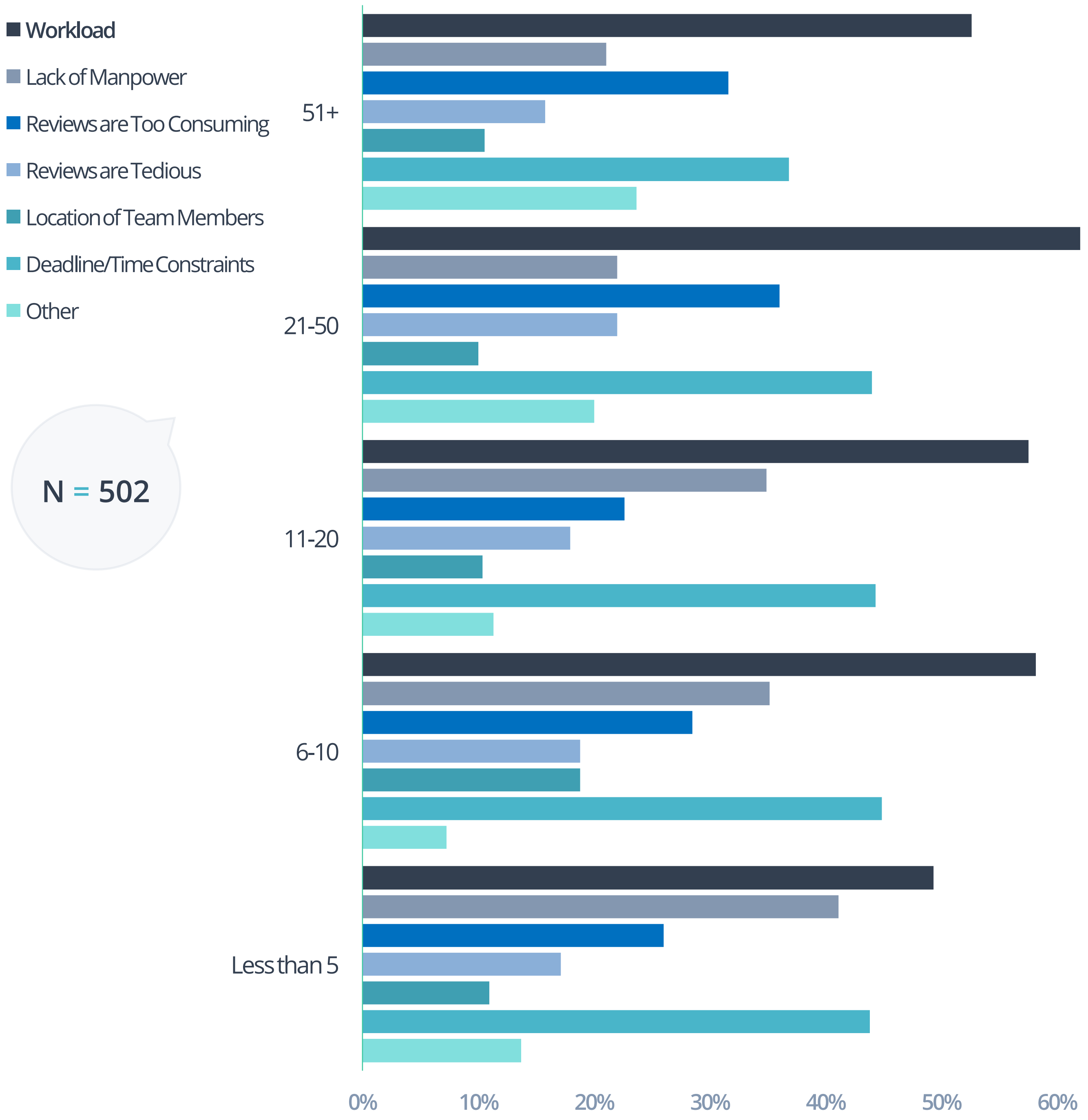
Similarly, time restraints and a lack of manpower are also looked at as primary roadblocks as teams struggle to allocate the necessary resources to conduct code reviews. Yet, more positive developments were revealed around manpower shortages in this year's survey. The surge in hiring has led to a drop of nearly 15% in those citing a lack of manpower from 2016 to 2017.

What obstacles prevent you from doing the level of code review that you desire?



As the below chart demonstrates, the leading obstacles to code review (workload and a lack of manpower) remain the same across organizations of all sizes.

What obstacles prevent you from doing the level of code review that you desire?



Section 3: Code Review Tools

The data from sections 1 and 2 reveal that most organizations can benefit from using a tool to assist with code review. There are several business drivers that determine the need for a code review tool, and choosing the right tool will depend on your specific software team's needs. In section 3 of the State of Code Review report, we take a closer look at how teams are choosing code review tools and the factors that go into selecting a tool.

Highlights

- Improving code quality is the number one reason respondents give to determine the need for a code review tool.
- 60% of respondents are using at least one code review tool; 24% are using Visual Studio.
- Other than code, requirements documents are the most commonly reviewed artifact type.
- Budget, managerial buy-in, and time constraints are the biggest obstacles to doing tool-assisted code review beyond workload considerations.

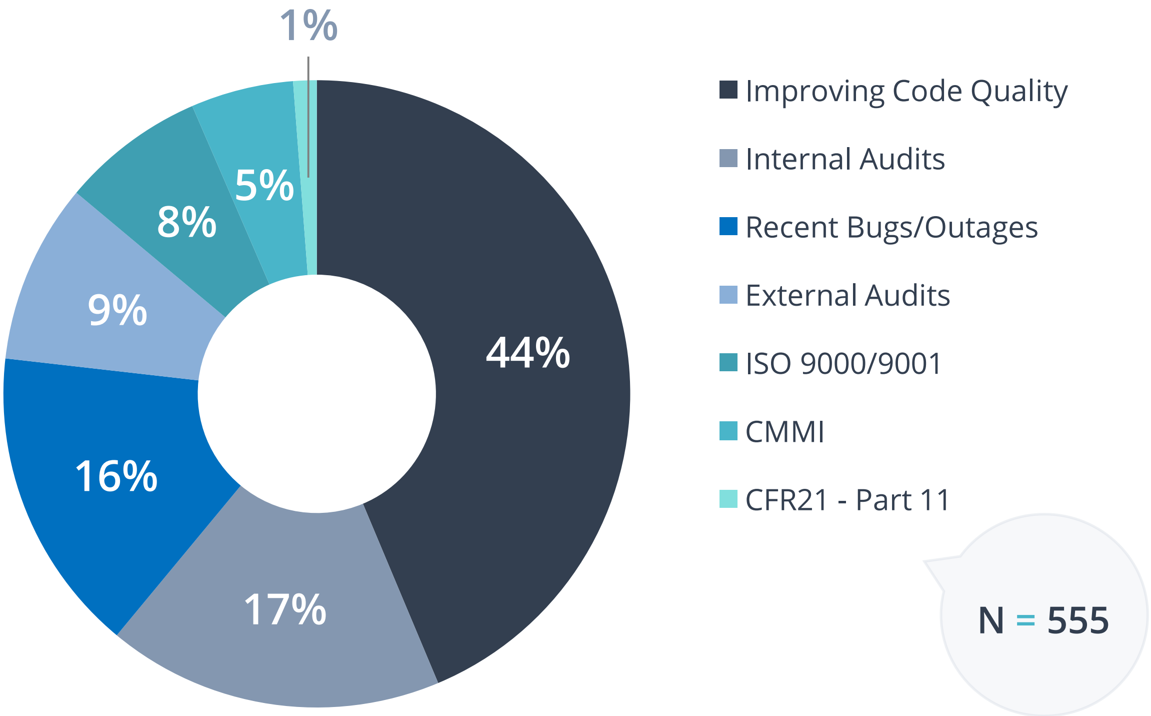
Improving code quality is the top reason for using a code review tool.

An overwhelming majority of respondents say that "improving code quality" is the number one business driver that determines their need for a code review tool. In fact, improved code quality outpaced the #2 driver – "internal audits" – by nearly three times. This is a significant jump from the 69% of respondents who cited code quality as the biggest business driver for using a tool and further underscores the widespread acceptance of code quality tools.

Rounding out the top four business drivers for using a tool were "external audits" and "finding bugs and product outages."

Among companies with fewer than 11 developers, finding bugs and product outages took the #2 spot, while internal audits were the #2 driver among development teams with 11 or more members.

What are the business drivers that determined your need for a code review tool?



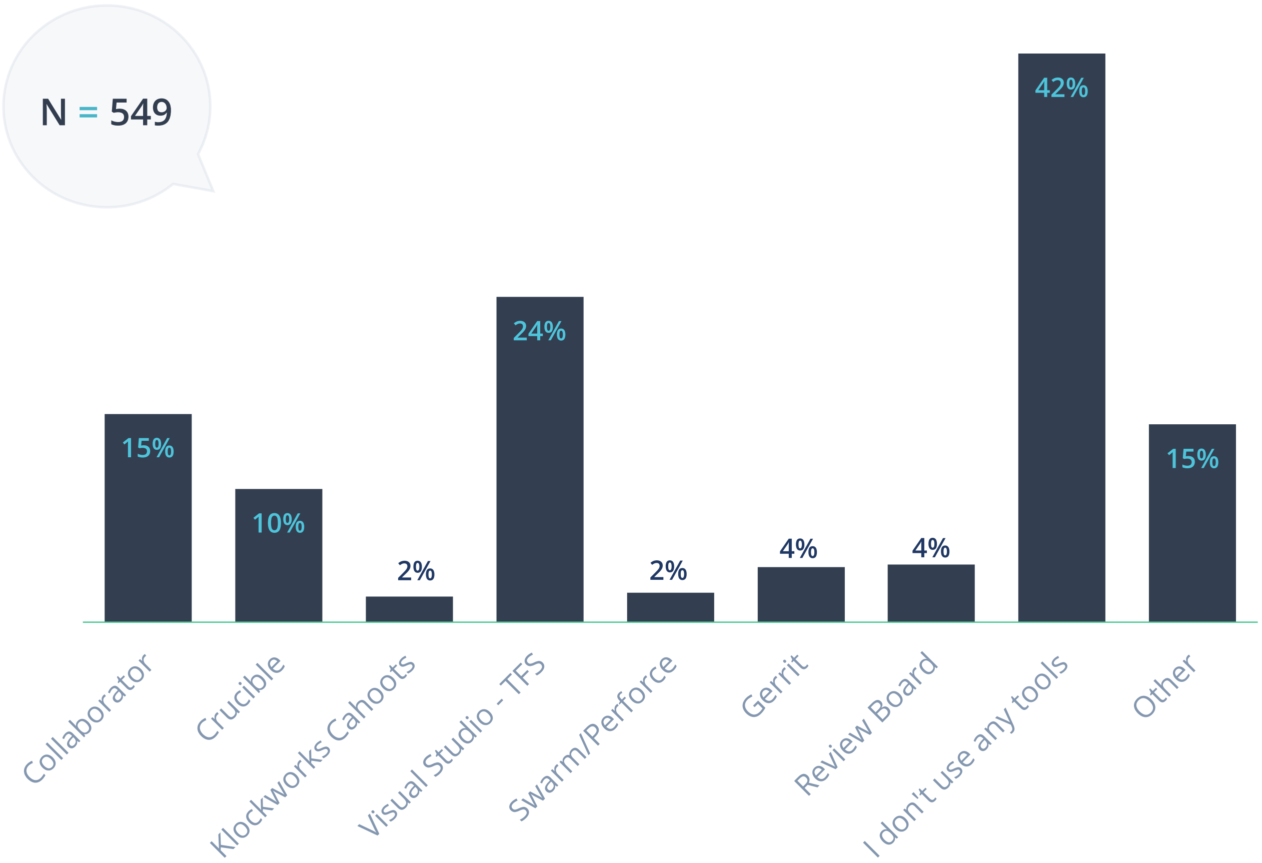
58% of respondents are using at least one tool for code review; Nearly 1 in 4 are using Visual Studio – TFS, and nearly 1 in 5 are using Collaborator.

The State of Code Review 2017 report is an industry-wide survey that was not limited to SmartBear customers. We are proud to see that Collaborator, our code review tool, was the most commonly used tool among organizations that are doing tool-assisted code review with a focused tool.

For those organizations that do not need best-of-breed code review functionality, they can now look within their TFS toolset for baseline code review features. This expanded use within TFS is the main driver behind the tool's code review usage climbing from 17.6% last year, to 24% this year.

In addition, we also found that 42% of teams are not using a tool for code reviews.

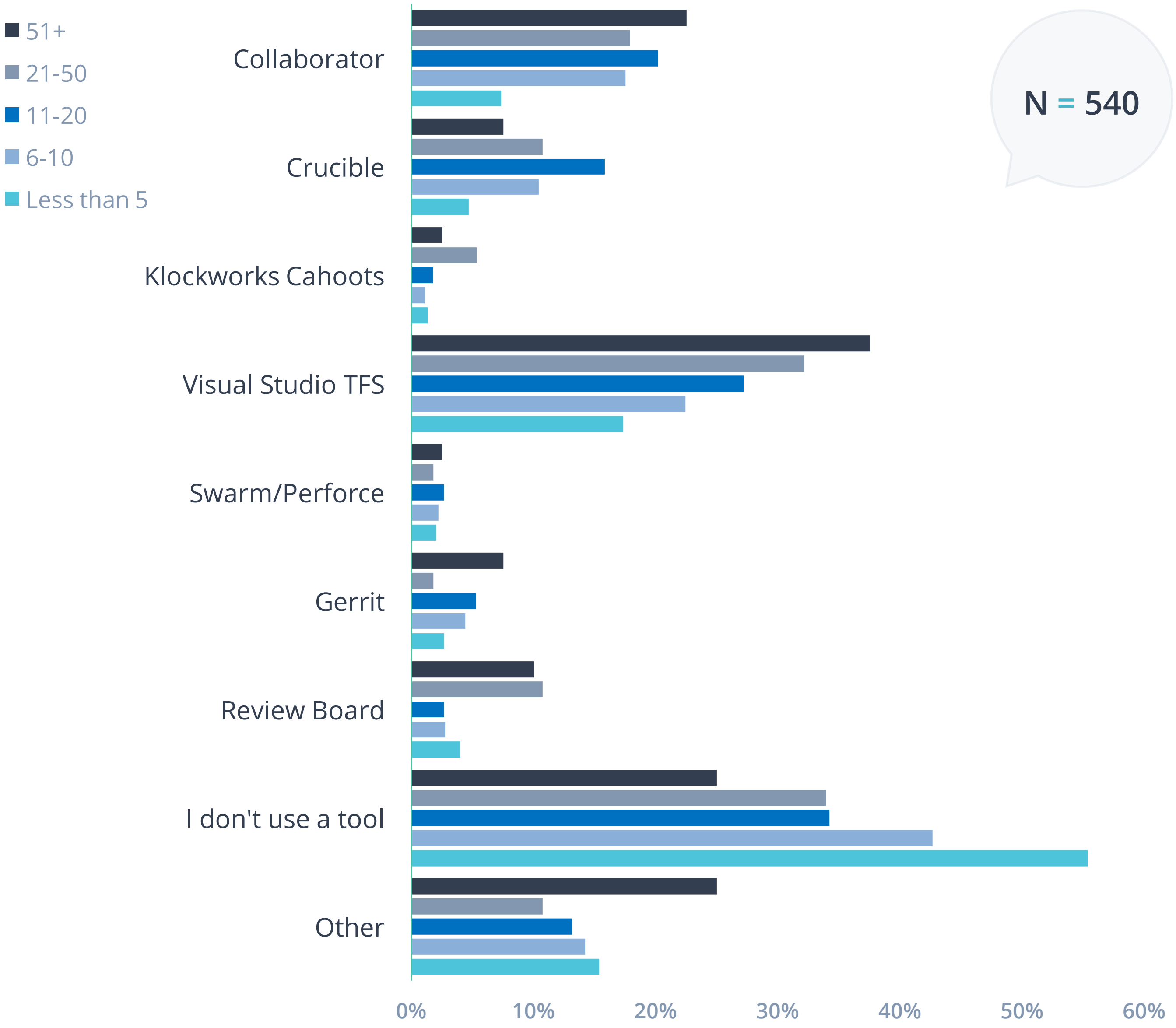
Within your department, do you currently use any of the following tools for code review?



90% of organizations with more than 50 developers ARE using a code review tool; 80% of development teams with fewer than 11 people are NOT currently using a tool for code review.

These findings reflect the fact that larger teams are performing more regular audits with a formal review process while smaller teams perform more ad-hoc reviews via informal collaboration.

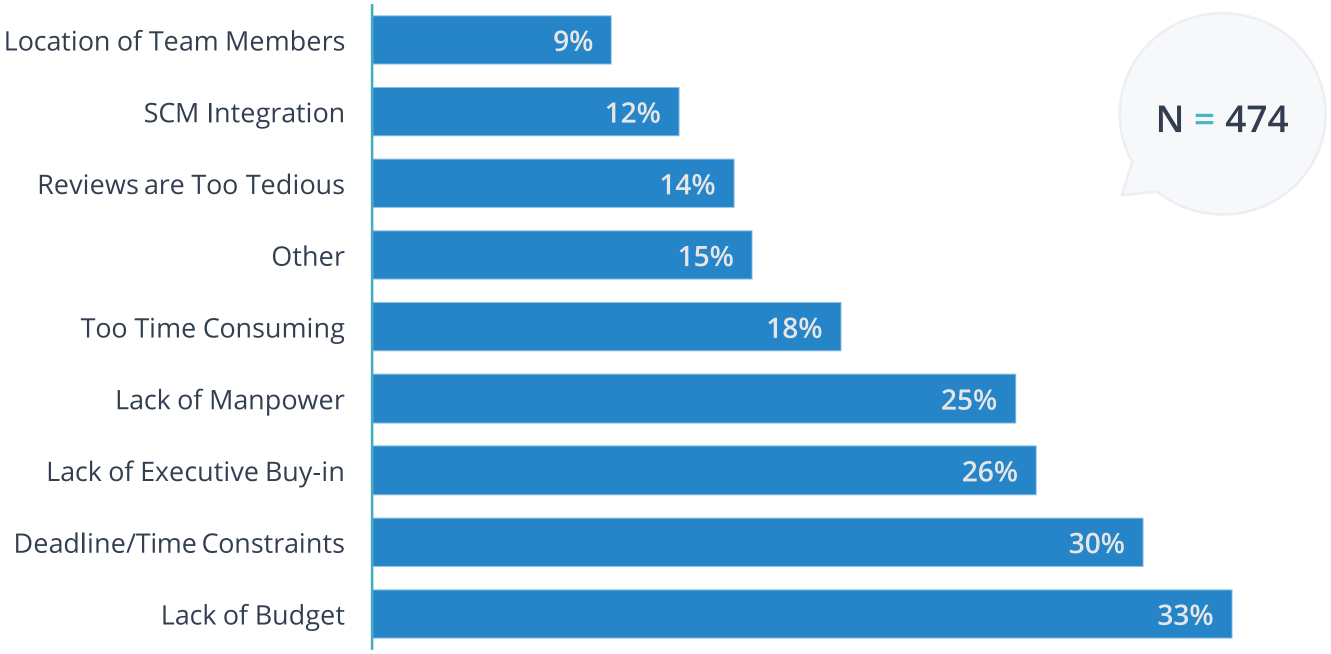
Code Review Tool Usage by Company Size.



Workload, budget, and time constraints are the biggest obstacles to doing tool-assisted code review.

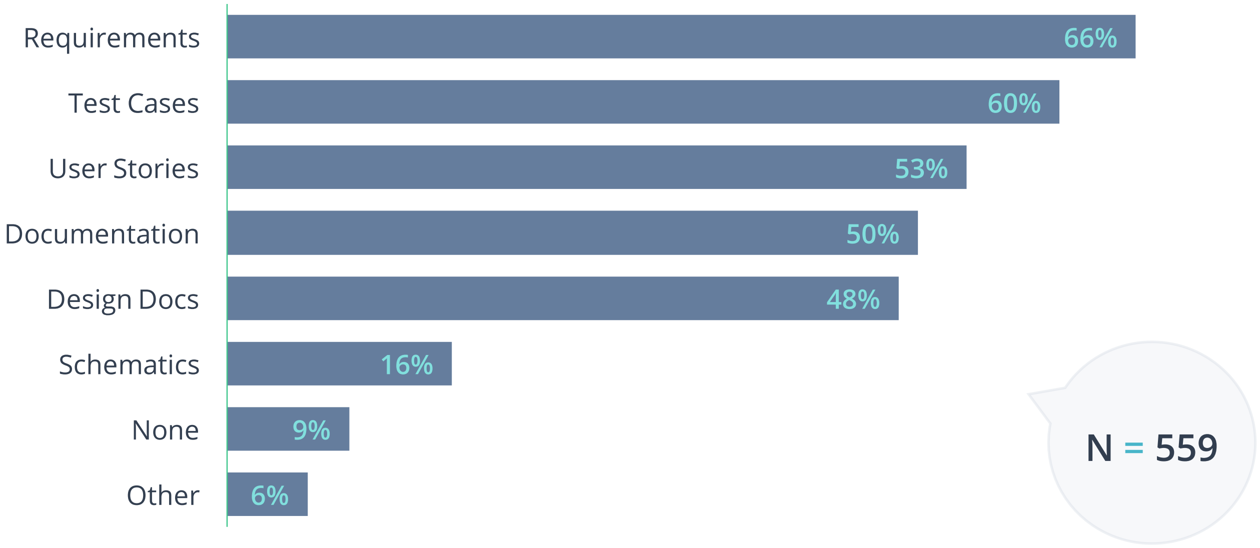
With approximately 40% of respondents not currently using a tool for code review, we wanted to better understand the factors that keep teams from adopting a code review tool. Beyond the common “workload constraints” answer, we delved deeper into the other obstacles that people found noteworthy. Respondents cited lack of budget (33%), management buy-in (25.5%), and manpower (24.7%) as the top blockers that prevent teams from adopting code review tools.

What obstacles keep you from doing tool-assisted code review?



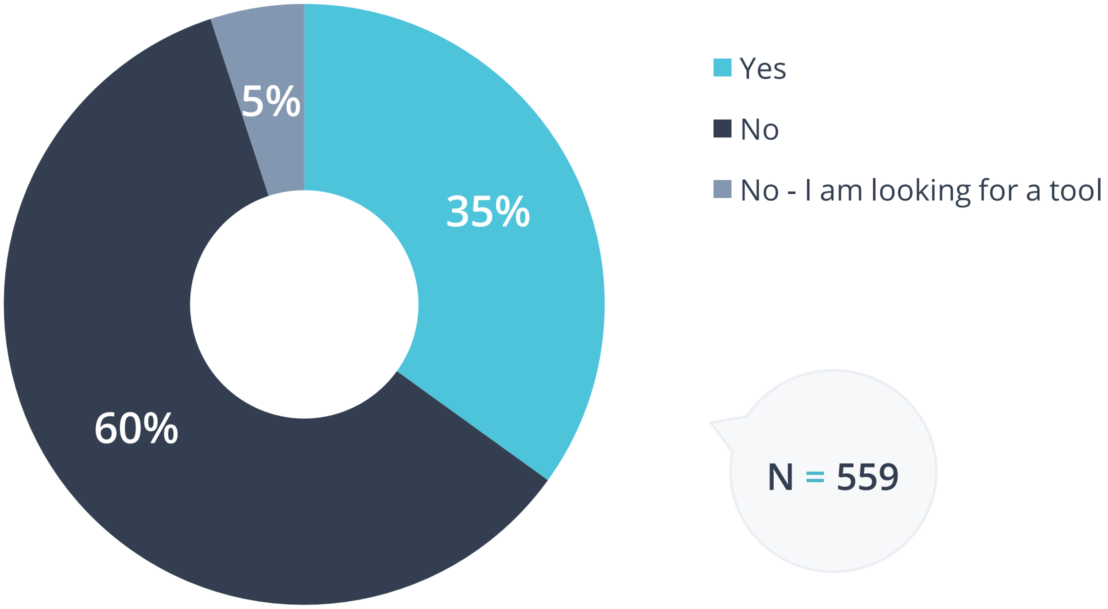
The most commonly reviewed artifact type is requirements documents (65.5%), followed closely by test cases (60%). User stories (53.3%), documentation (49.8%), and Design documents (48.4%) round out the top 5 most-reviewed artifact types

We review the following types of artifacts:



While just 35% of respondents use a tool to review their artifacts, only 4.9% stated that they are in searching for a tool to use for artifact review.

Are you using a tool to review these artifacts?



Section 4: Software Tools & Integrations

Code review can be counted among numerous critical processes for developing and maintaining software quality. As such, software teams rely upon an array of tools and systems during software development. In section 4 of the State of Code Review 2017 report, we examine several trends within the software industry related to the adoption of software tools and systems.

Highlights

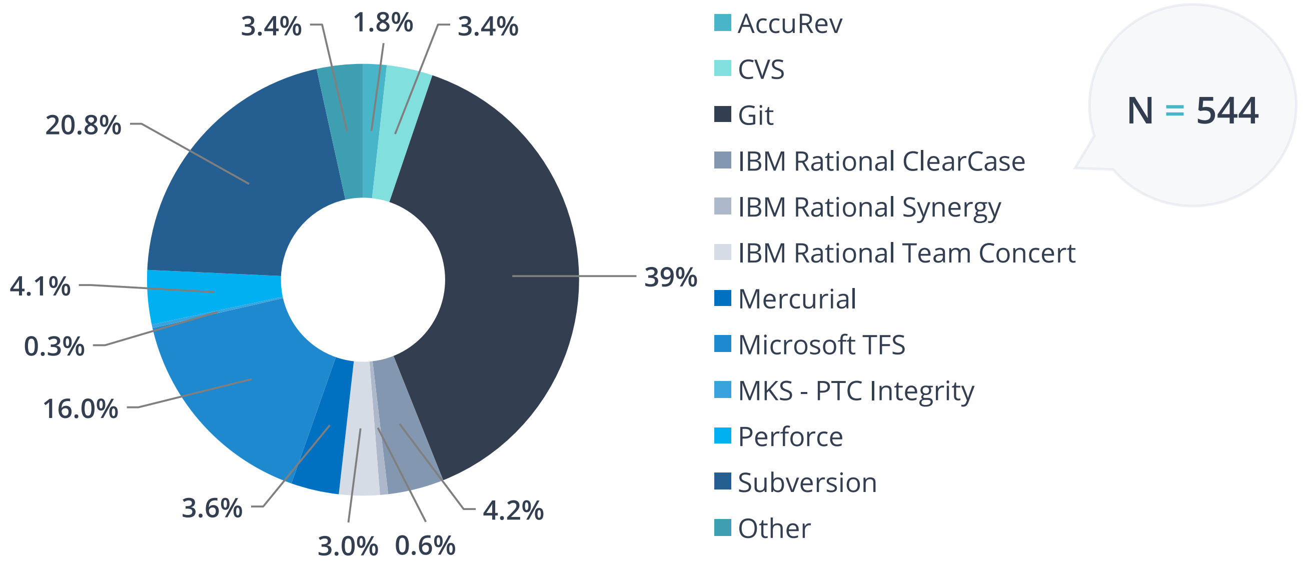
- Git is the most commonly used software configuration management tool (SCM), with 57.3% of respondents now using Git.
- 61% of respondents are not using a repository management tool.
- More than 70% of respondents are not planning on introducing a repository management tool in the next three months.
- 49% of respondents are using Visual Studio as their integrated development environment; 45% are using Eclipse.
- JIRA is the most commonly used tool for bug tracking (57.7%). JIRA is also the top tool for requirements management (34%).

Git is the most commonly used software configuration management tool, with 57.3% of teams using it.

Fifty-seven percent of respondents cited Git as their choice for an SCM tool, which represents a 14-point increase over 2016. We attribute the increase to GitHub staffing up and making a significant sales push. Strong growth was also experienced by BitBucket and GitLab. Overall, growth in software configuration management tool purchases was significant, as evidenced by GitHub's annual recurring revenue from large clients increasing from $25 million to $70 million at one point in 2016.

Almost one-third of development teams are using Subversion, and nearly one quarter are using Microsoft TFS.

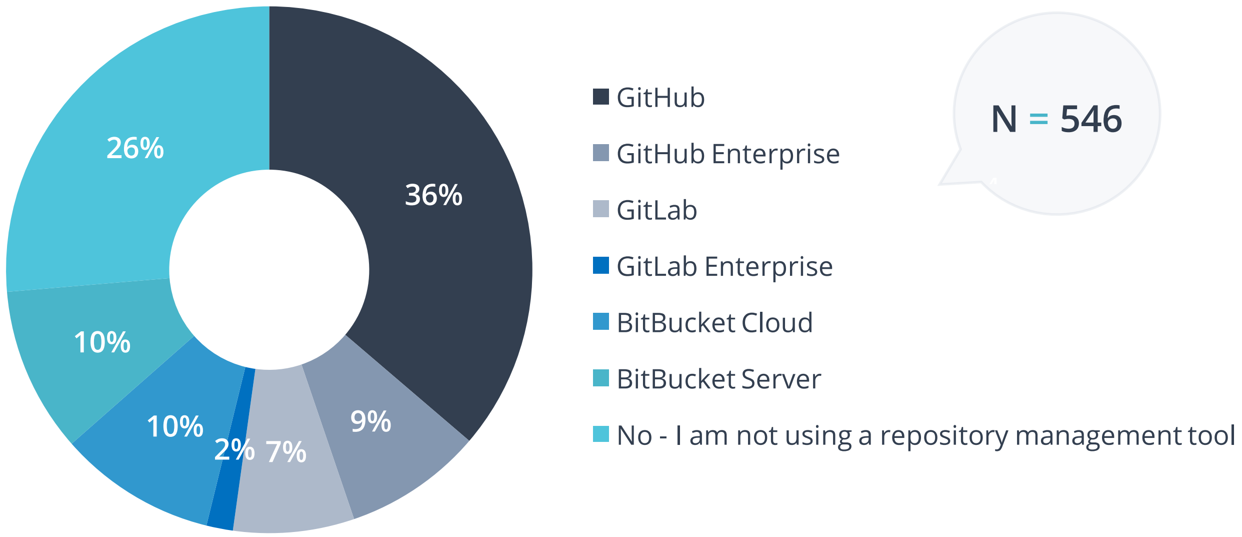
Which software configuration management tool (SCM) do you or your company currently use?



71% of respondents are not using a repository management tool; 1 in 5 are using GitHub.

71% of respondents are not using any of the well-known repository management tools. Of those who are using a repository management tool, more are using GitHub than any other tool. Users cited GitHub at a 3-4 times higher rate than the second-most cited tool, Bitbucket Server.

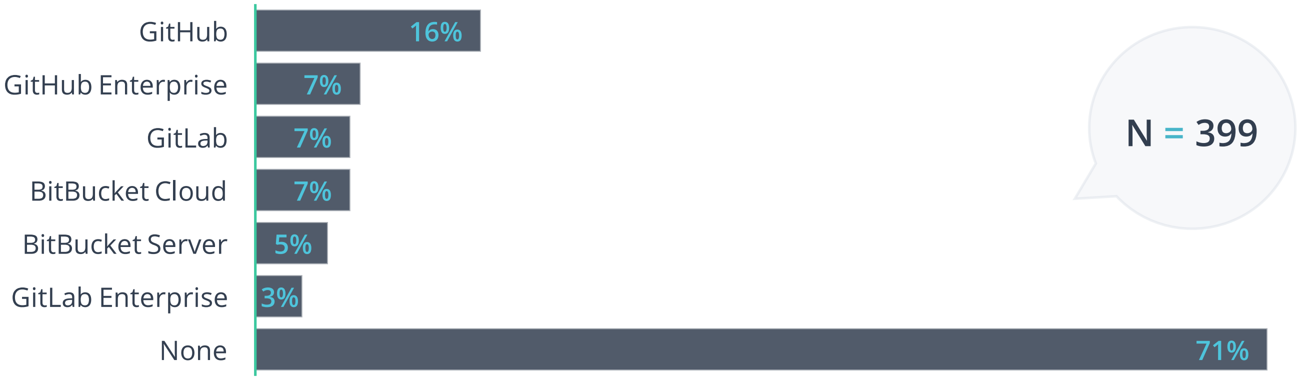
Which of the following repository management tools are you currently using?



More than 70% of respondents are not planning on introducing a repository management system in the next three months.

Of those respondents that are planning on introducing a repository management system, a majority are planning on implementing GitHub or GitHub Enterprise.

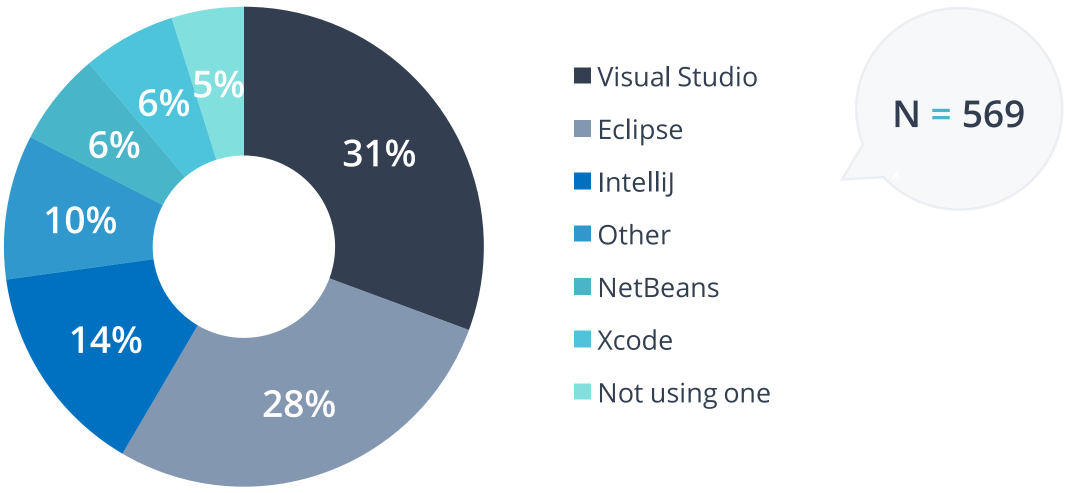
Are you planning on introducing any repository management tools in the next 12 months?



50% of respondents are using Visual Studio as their integrated development environment; 45% are using Eclipse.

A vast majority of respondents are using an integrated development environment (92%) — with half using Visual Studio and nearly half using Eclipse. The use of these tools is definitively on the rise, as Visual Studio (up 26%) and Eclipse (up 125%) rose far above last year's numbers. More than 1 in 5 respondents are using IntelliJ.

Which integrated development environment are you currently using?

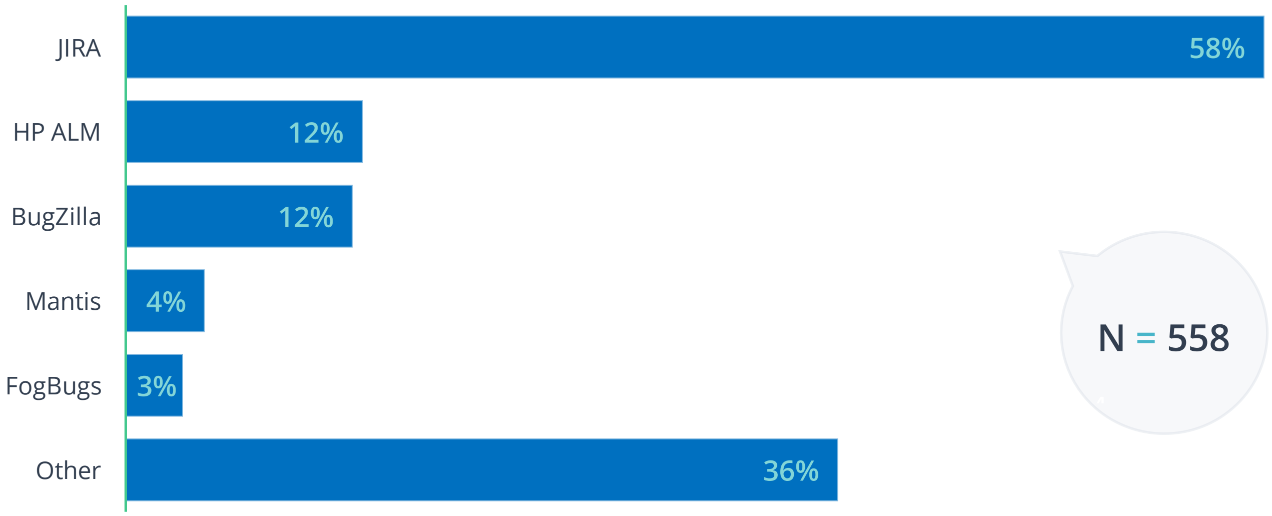


JIRA is the most commonly used tool for bug tracking, with 57.7% of respondents using JIRA.

JIRA outpaced all other bug tracking tools by at least a 4:1 margin. JIRA's numbers jumped 16 percentage points from 2016 to 2017, and we attribute this to Atlassian going public in late 2015, providing the company with the resources to perform marketing, sales, and service at greater scale.

For organizations that aren't using JIRA, there is a lot of variation in how they are tracking bugs, with no clear runner up to JIRA.

What tool are you using for bug tracking?

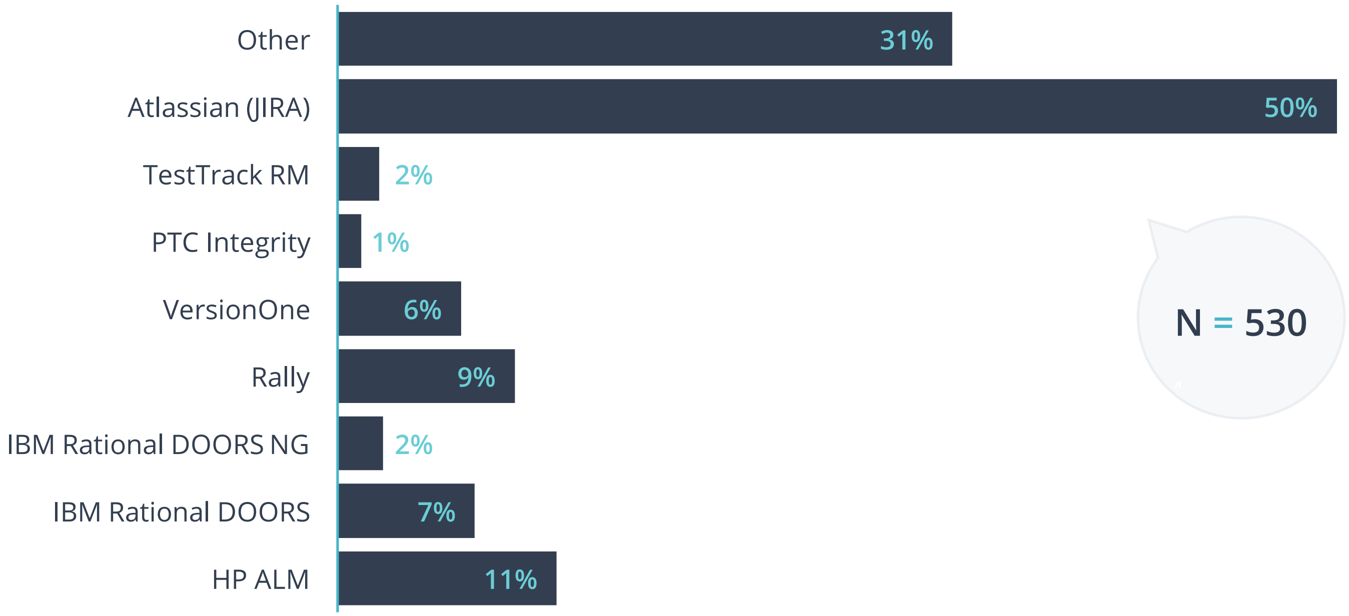


Atlassian (JIRA) is the most commonly used tool for requirements management, with 50% of organizations using JIRA.

Similar to what we saw with bug tracking tools, JIRA dominates and saw a percentage jump of 16 points between the 2016 and 2017 surveys. There is also a good amount of variation in the tools organizations are using for requirements management. While JIRA is being used by half of organizations, there were no other clear standout tools for requirements management.

31% of respondents cited a tool other than the 8 most popular tools on the market for requirements management.

What tool are you using for requirements management?



Section 5: Software Teams

The State of Code Review 2017 report includes responses from software developers, testers, IT/operations professionals, and business leaders representing more than 30 different industries. Participants in the survey work on software teams of all sizes, from less than 5 employees, to up to more than 51 employees. They also work for companies ranging from small businesses, with less than 25 employees, to enterprise organizations, with 10,000 employees or more.

Highlights

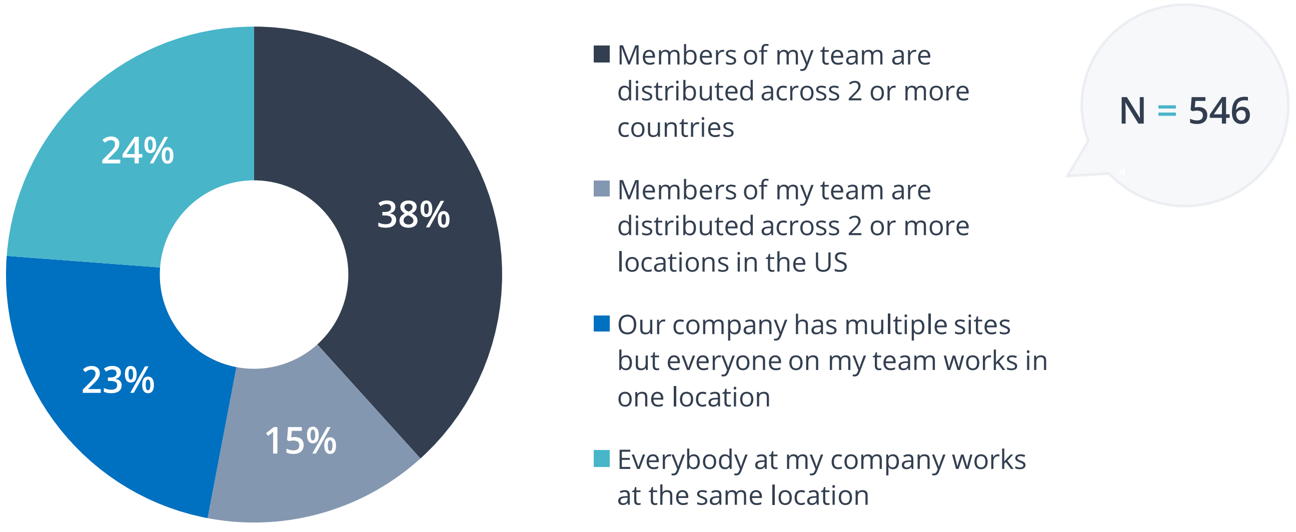
1. More than half of teams are distributed across multiple locations; 38% are international.
2. 45% of respondents work in companies with more than 10,000 employees; 1 in 4 work in companies with less than 100.
3. 78% of respondents work on development teams with 20 members or less.
4. More than 30 industries are represented in the State of Code Review 2017 report.
5. 48% of survey respondents are developers; 17% are testers.

More than half of teams are distributed across multiple locations; 38% are international.

Looking at how teams are distributed, we found that a majority of them are now working across multiple locations. Of those multi-location teams, a majority are collaborating across two or more countries.

Just 15% of organizations have teams that work in more than one US location.

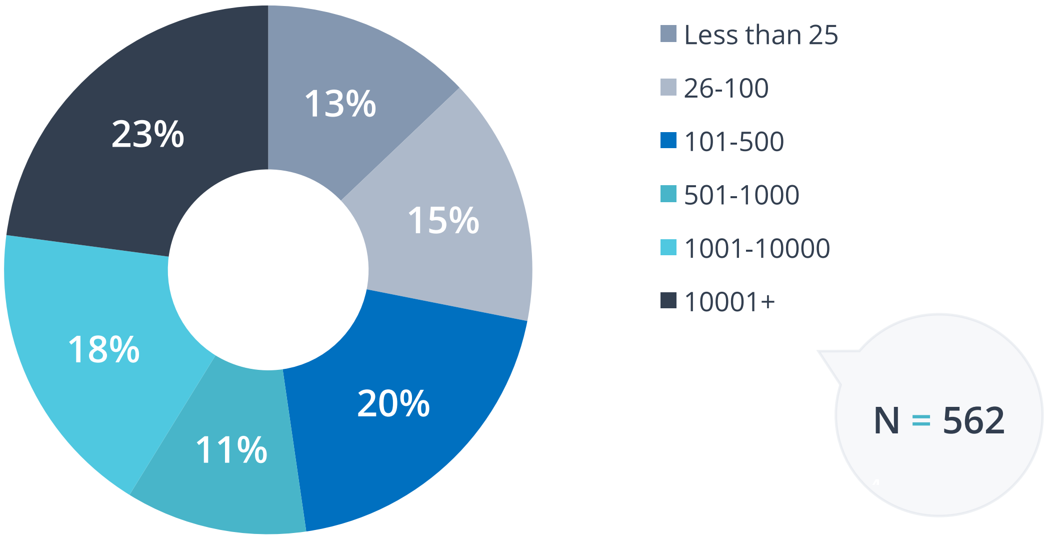
Which best describes your development team?



23% of respondents work in companies with more than 10,000 employees; 28% work in companies with less than 100.

The State of Code Review 2017 report includes responses from software professionals working in small, medium, and enterprise organizations. Where last year 45% of survey respondents worked at companies with more than 10,00 employees, this year's pool of respondents had 23% from such large companies. By having all company sizes reflected in a more balanced survey pool, this year's data provides a more broadly representative data set.

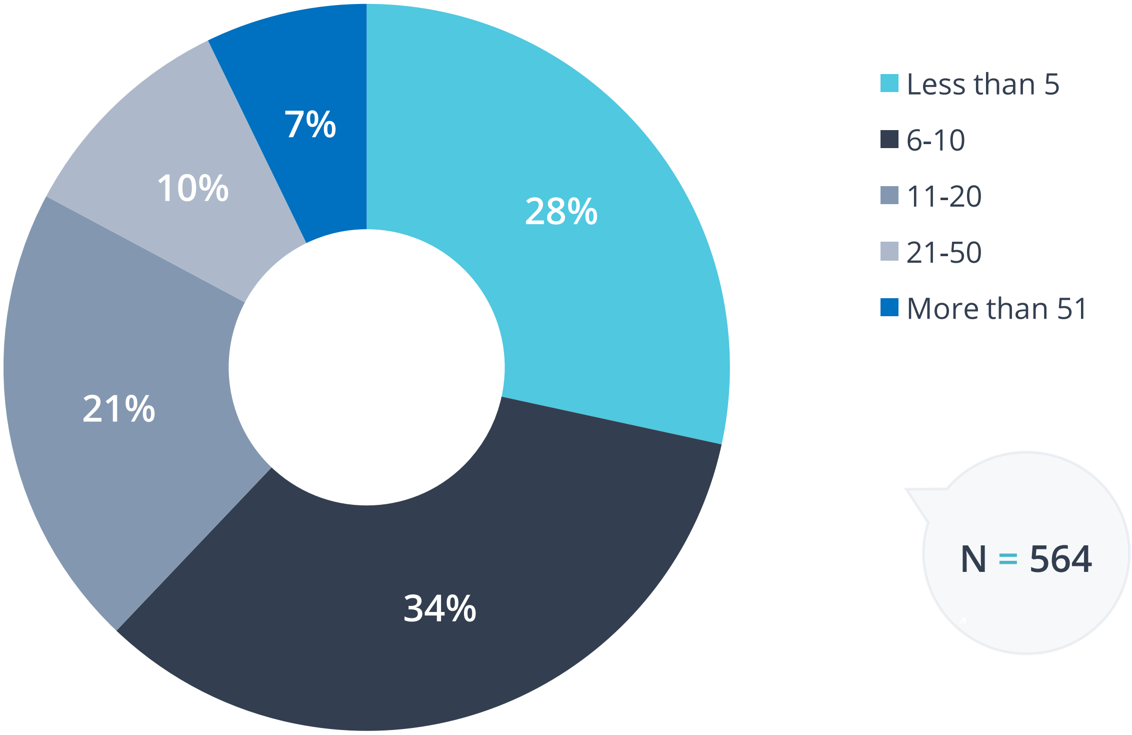
What is the total employee size of your company?



83% of respondents work on development teams with 20 members or less.

Respondents in the State of Code Review 2017 survey work primarily on smaller development teams — with nearly 3 in 10 on teams having fewer than 5 employees. As companies pursue more agile approaches to software development, team sizes are shrinking (even at large companies).[1] This explains why only 17% of respondents work on development teams of 21 or more members.

What is the size of the development team you are on?



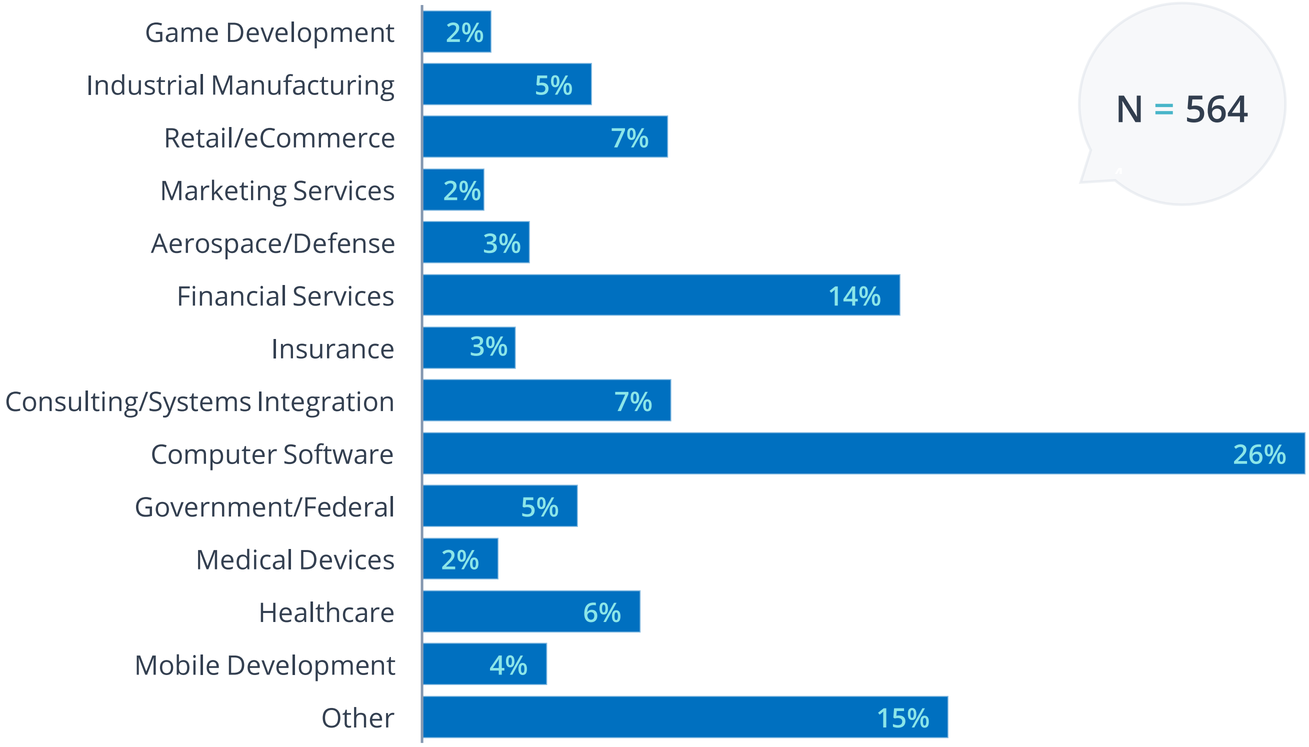
N = 564

More than 30 industries are represented in the State of Code Review 2017 report.

Computer software and financial services are the industries with the largest representation in the State of Code Review 2017 report, followed by consulting/systems integration, retail/ecommerce, and healthcare.

[1] Schwaber, Ken and Sutherland, Jeff (2014). "The Scrum Guide," Scrum.org and Scruminc. Copyright 2014. Referenced online April 5, 2017 from the following source: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf

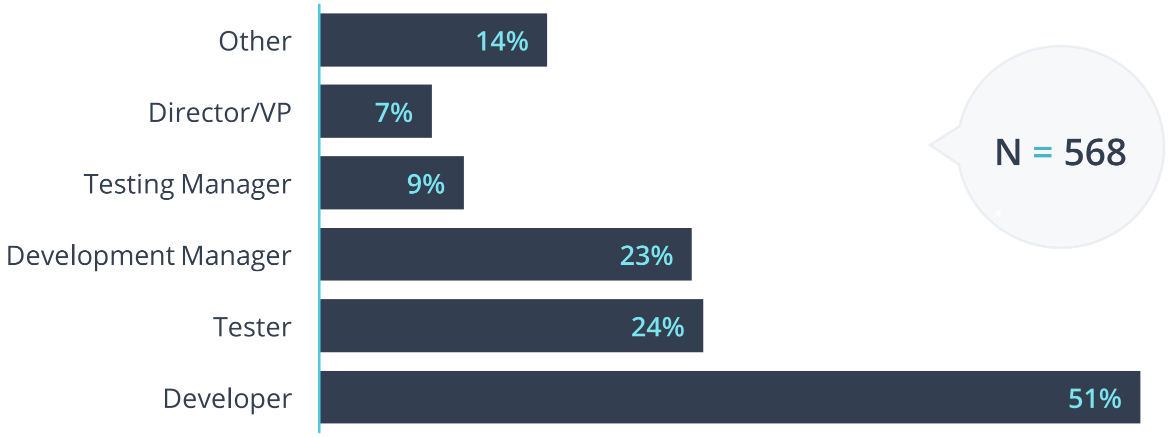
What industry do you work in?



51% of survey respondents are developers; 24% are testers.

Development and QA combined for 79% of the roles represented in the State of Code Review 2017 report.

What best describes your current role?



SMARTBEAR

The Leader in Software Quality Tools for **Teams**

6.5M+	20K+	194
Users	Companies	Countries

We create the software tools that development, testing, and operations teams use to deliver the highest quality and best performing software possible, shipped at seemingly impossible velocities.

With products for API design and documentation, API and UI level testing, code review, and monitoring across APIs, web, mobile, and desktop applications, we equip every member of your team with tools to ensure quality at every stage of the software cycle.

SmartBear.com **@SmartBear**