The State of Code Review

Trends and Insights into Collaborative Software Development

2018

SMARTBEAR

Content

Preface

This survey was designed to establish benchmarks for the software industry concerning the ways in which teams and organizations are developing high quality software in 2018. The structure of this year's report closely mirrors the structure of our past editions in order to provide a meaningful year-over-year comparison and uncover significant trends around code quality and development approaches. This report covers the following topics:

- Perceptions on Code Quality Practices
- Common Approaches to Code Review
- Tools & Systems Across Development
- Reception to Trends and Changes to Teams

Methodology

SmartBear Software conducted a global online survey over the course of seven weeks, from February to April 2018. The findings presented are based upon aggregated responses from more than 1100 software developers, testers, IT/operations professionals, and business leaders representing more than 35 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

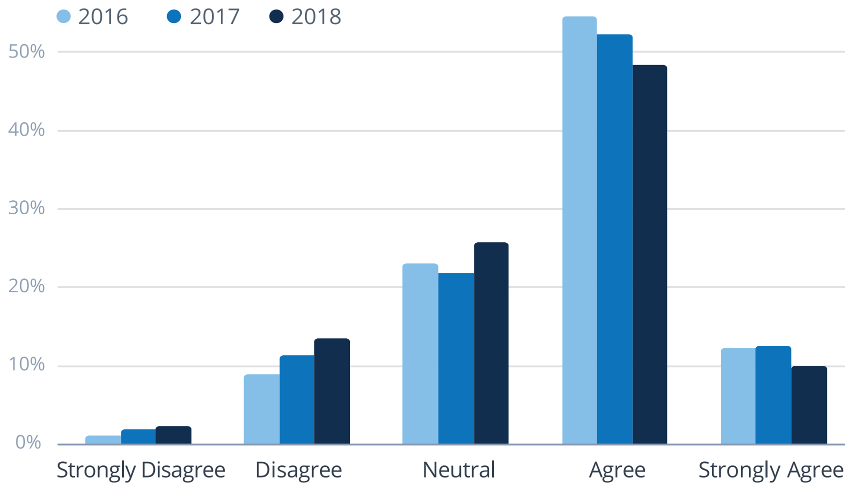
Code Quality in 2018

While this report covers a number of subtopics around code quality, these are some of the most interesting insights from this year's State of Code Review report.

While roughly two-thirds of respondents are satisfied with the quality of their software, there is a general downward trend since 2016.

This change, though not seismic, is likely indicative of the increasing challenges that modern development teams face. From cybersecurity concerns to the rise of the API economy, software complexity is on the rise.

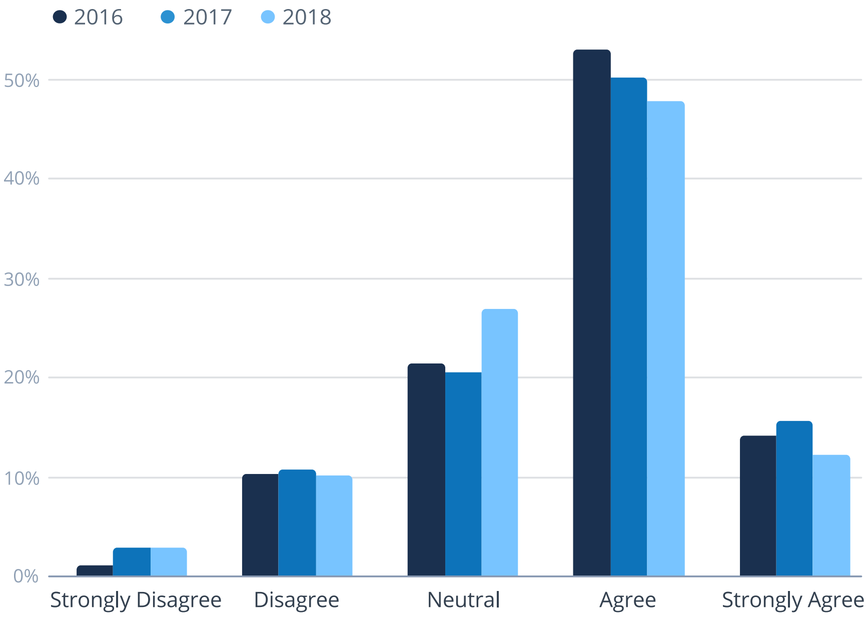
I am satisfied with the overall quality of software I help deliver.



n = 537   n = 566   n = 849

Another indicator that software development is becoming more challenging is a similar downward trend with regards to teams meeting their release deadlines. We saw a decrease in number of respondents who regularly release code on time.

My company is regularly able to get releases out on time.



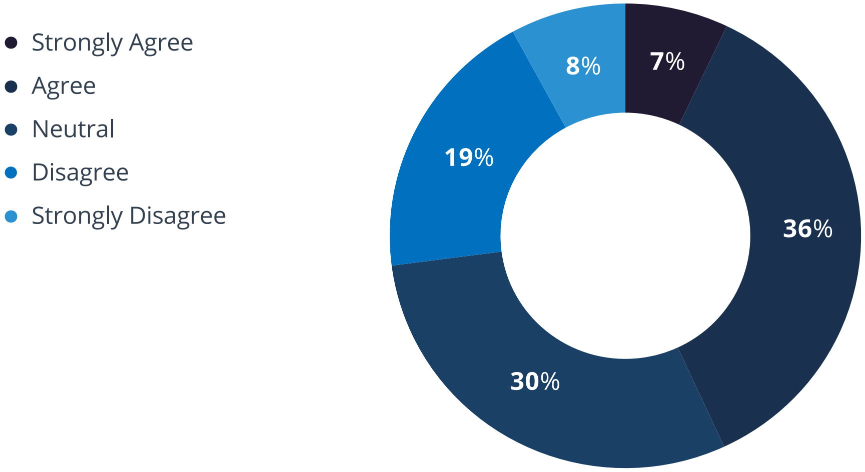
n = 537 **n = 565** **n = 851**

It is worth noting that this chart does not speak at all to the frequency of releases, but just deadline success broadly. As teams become more agile, it is possible that an increased release frequency could impact this measurement.

Only 43% of respondents are satisfied or very satisfied with their current code review process.

This year, we added the following question on code review process satisfaction to our survey:

I am satisfied with my team's current code review process.



n = 941

Since code reviews can seem like a chore, it is reasonable that so many developers would be indifferent or neutral to their satisfaction with their process. For many respondents, they might not have much control over their own process or their process might be less formally defined.

More respondents are at least somewhat satisfied with their code review process (**43%**) than somewhat dissatisfied (**27%**). Later in this report, we will reference these two cohorts to better understand how code review process satisfaction impacts other development attributes.

For the 3rd straight year, respondents identified code review as the number #1 way to improve code quality.

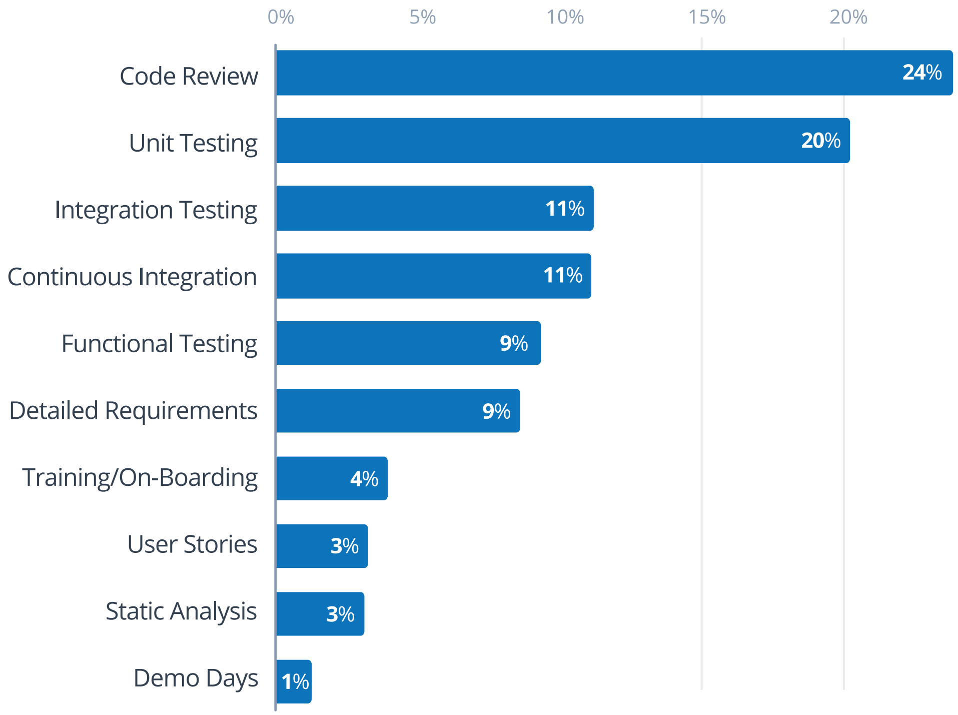
Since 2016, respondents to our annual report have overwhelmingly identified code review as the top practice for improving code quality. Each year, unit testing has followed as the 2nd highest ranked choice.

We compared overall software satisfaction between two cohorts, unsatisfied with their code review process and satisfied with their code review process. The perception of code reviews being a key to better software quality is validated by this year's data.

What you can see in this chart below is the correlation between code review satisfaction and satisfaction in overall software quality. 81% of respondents who were satisfied with their code review process were also satisfied with the overall quality of their software. Respondents who were not satisfied with their code review process were half as likely to be satisfied in their overall software quality, with only 40% respectively.

?

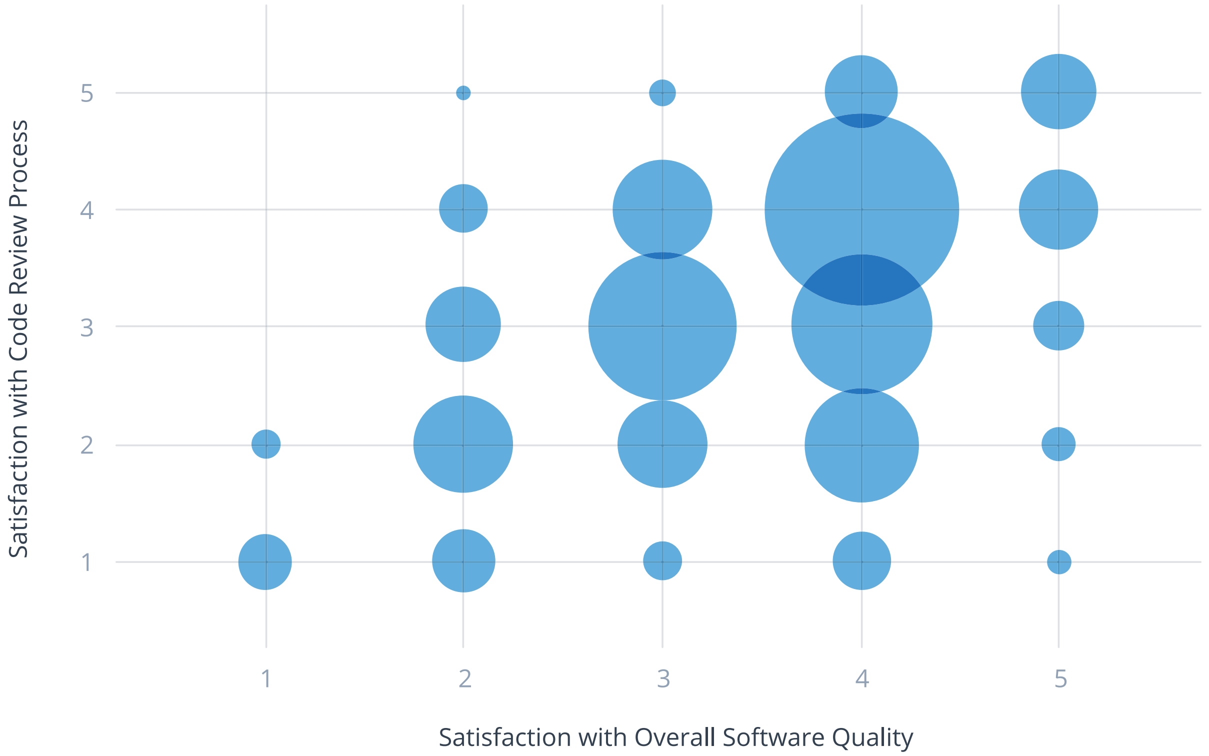
What is the number #1 thing a company can do to improve code quality?



n = 856

Of all the measures and attributes we surveyed, satisfaction with one's code review process was the most likely indicator of satisfaction with overall software quality. Over the course of this report, we will show year-over-year changes in code review practices and identify common traits that contribute to a better peer review approach.

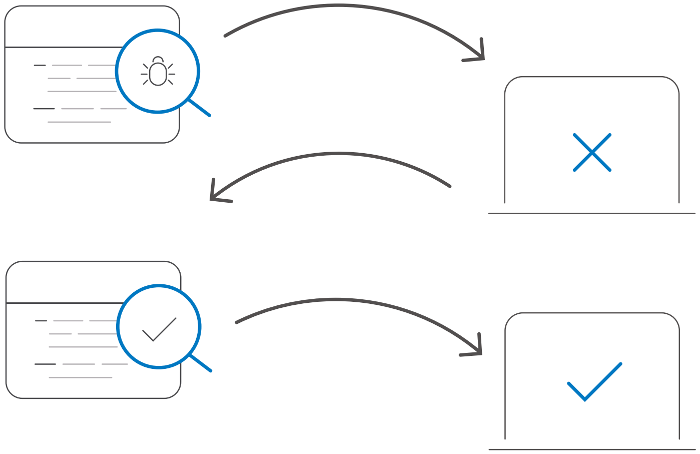
Correlation between code review process satisfaction and code quality satisfaction



n = 848

Section 1: The Code Review Process

Whether development teams find their process through natural iterations or intentional requirements, there are countless configurations of different workflows, frequencies, toolsets, and behaviors. In this section, we will look at what teams are doing today.



Highlights

- 64% of respondents are participating in code reviews on at least a weekly basis, with 32% reviewing code daily.
- 48% of respondents are participating in "ad hoc", or over the shoulder reviews, on at least a weekly basis, with 18% reviewing daily.
- 39% of respondents are participating in tool-based code reviews on at least a weekly basis, with 21% reviewing daily.
- 23% of respondents are participating in meeting-based reviews on at least a weekly basis, with 5% reviewing daily.

These three methods (ad hoc, meeting-based, and tool-based) are the most common approaches to code review. This chart shows the distribution of review type and frequency in this year's survey:

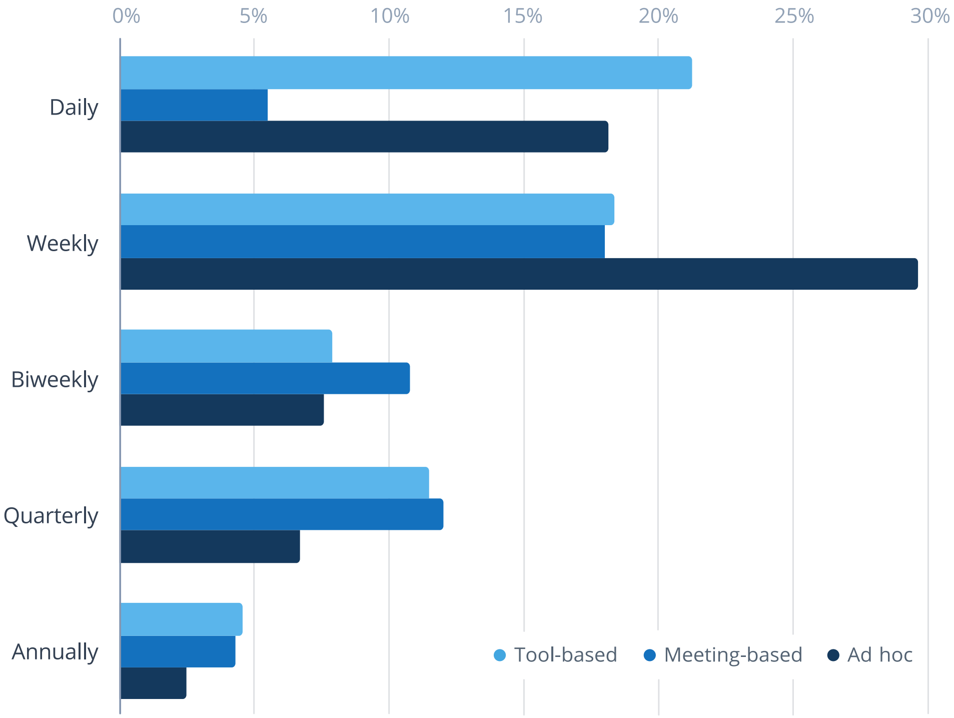
Ad Hoc Reviews

Over the last three years, there was been a steady increase in the frequency and prevalence of ad hoc reviews. These are some of the key metrics in this year's survey that show that change:

- 18% of respondents are conducting ad hoc reviews on a daily basis. This has been increasing steadily, with that figure at 16% in 2017 and 14% in 2016.
- 56% of respondents are conducting ad hoc reviews on at least a biweekly basis. This number has also been on the rise compared to 53% in 2017 and 43% in 2016.
- 22% of respondents are not conducting ad hoc reviews at all. This is down respectively from 25% and 28% in the two years prior.

n = 1124

How often do you participate in these common code review approaches?



Tool-Based Reviews

Over the last three years, there hasn't been much change to the frequency and prevalence of tool-based code reviews. The only significant change was a 6% increase from 2017 in respondents conducting some form of tool-based reviews.

- 21% of respondents are conducting tool-based reviews on a daily basis. In 2017 and 2016, that figure was 21% and 23% respectively.
- 47% of respondents are conducting tool-based reviews on at least a biweekly basis. In 2017 and 2016, that figure was 48% and 44%.
- 66% of respondents are conducting tool-based reviews, which is a slight increase from 2017 (60%) and 2016 (63%).

Meeting-Based Reviews

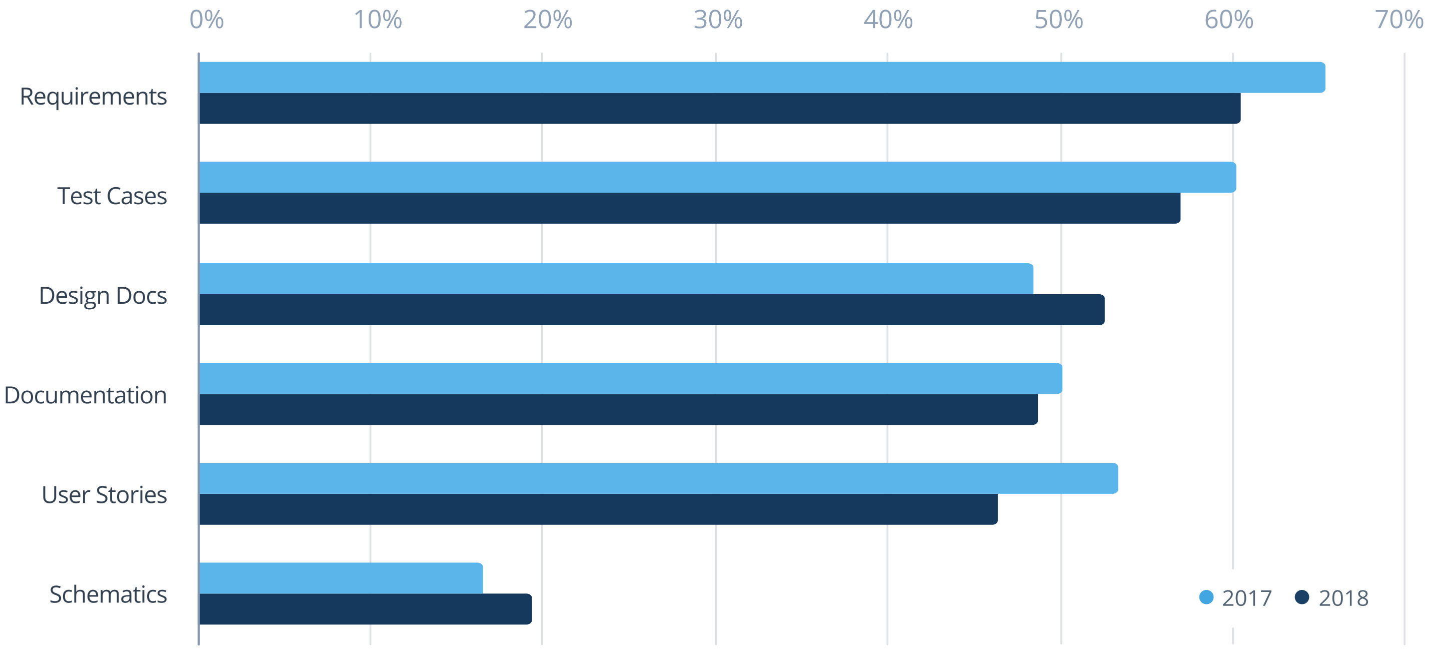
Over the last three years, respondents are increasingly participating in more meeting-based reviews. Despite teams being more dispersed than ever, new communication platforms like Slack, Stride, Flock, and GChat are making it easier than ever for teams to quickly connect. Meetings that might have required participants to be in the same room ten years ago are now easily setup with video chat tools like Zoom and GoToMeeting.

- 5% of respondents are conducting meeting-based reviews on a daily basis. In 2017 and 2016, that number was similar at 6% and 4% respectively.
- 34% of respondents are conducting meeting-based reviews on at least a biweekly basis. This has been steady rising in the last three years, with 30% in 2017 and 21% in 2016.
- 65% of respondents are conducting meeting-based reviews. This number is up slightly from 2017 (60%) and 2016 (63%).

Document Reviews

In addition to asking about peer code reviews, we also track how teams are reviewing documents year over year. Last year, we found that 91% of respondents were participating in the review of some kind of document. This chart shows the kinds of documents most commonly reviewed and how it compares to last year's data.

Which of the following artifacts do you review, if any?



n = 561 n = 1102

Despite 90% of respondents saying that they are participating in document reviews, only 35% said that they use a tool to review artifacts. This percentage remained flat year-over-year. Most teams are likely conducting these document reviews through general communication means, whether that is in a meeting, over chat, or over email.

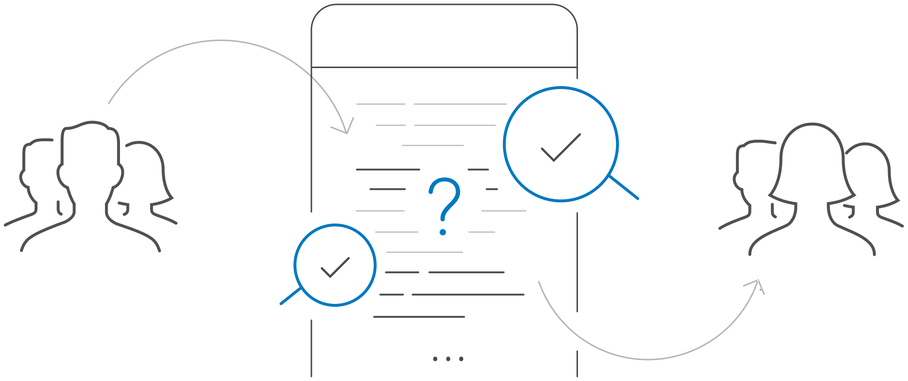
Cohorted by overall code satisfaction, 47% of respondents satisfied with their code review process had a tool for reviewing artifacts. That number was only 21% for respondents in the dissatisfied code review cohort.

Additionally, the satisfied code review cohort responded that they reviewed on average 2.1 types of documents. The dissatisfied cohort responded that they reviewed on average 1.7 types of documents.

This data suggests that teams who conduct more document reviews and have a tool-based review capability are more likely to have a more satisfactory code review process and therefore higher code quality satisfaction. One reason for this could be the valuable transfer of knowledge that happens during reviews of requirements and test cases, leading to more harmonious development.

Section 2: Perceptions on Code Review

We know that a satisfactory code review process is critical to producing satisfactory overall code quality, so we asked developers what they thought about code review. This section looks at common benefits, obstacles, and business drivers around their process.



Highlights

- 73% of respondents say that they know what they are supposed to be looking for when they review a teammate's code.
- 55% of respondents chose "Workload" as the top obstacle to conducting code reviews, followed by "Deadline / Time Constraints" at 42%.
- 90% of respondents say that the most important benefit of code review is "Improved Software Quality", followed by "Sharing Knowledge Across the Team" at 73%.

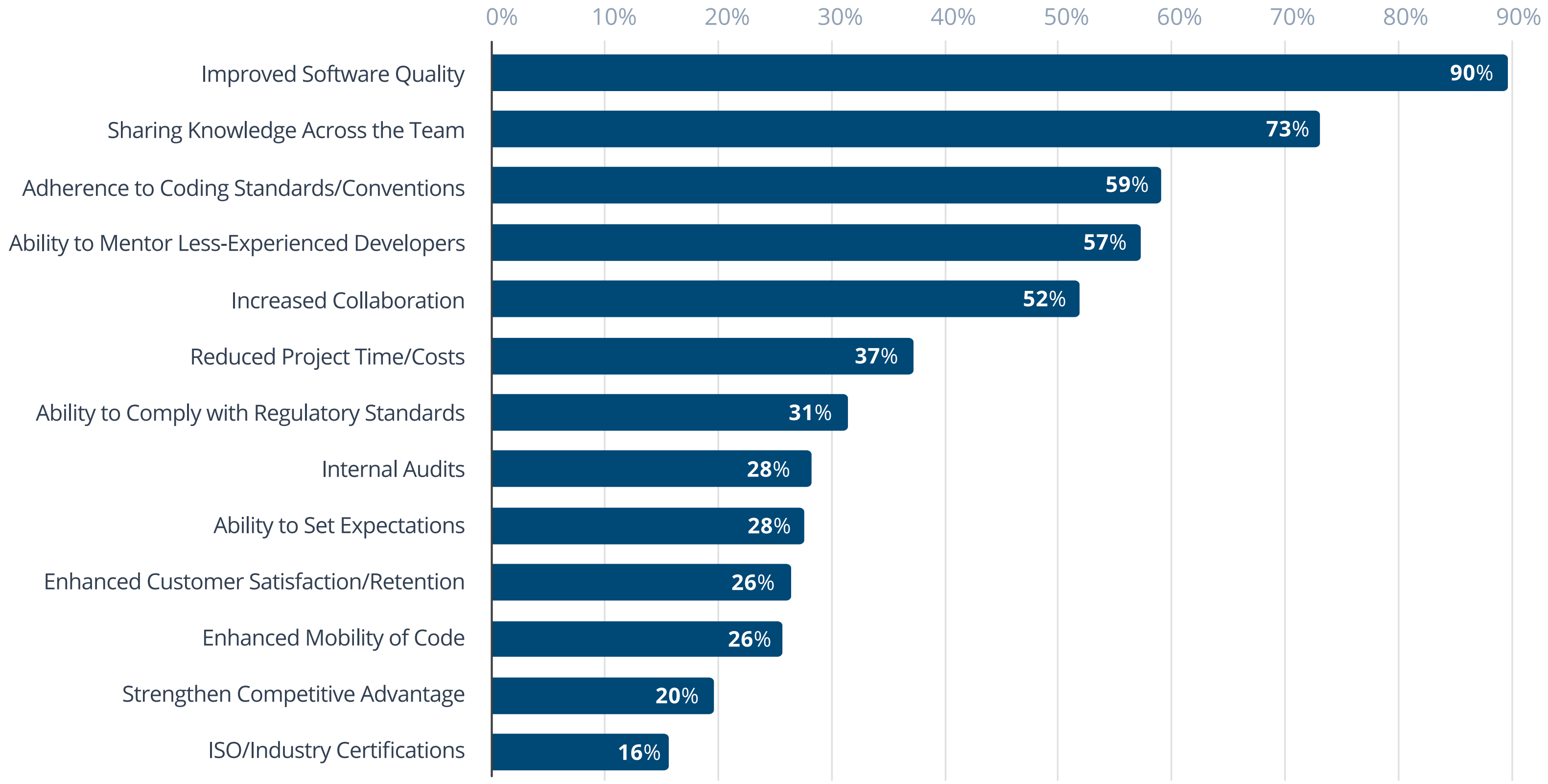
The Top Benefits of Code Review

Just as there are a wide variety of code review workflows and approaches, there are also different benefits that teams can derive from reviews. Finding bugs and defects is the core function, but the process itself can improve general communication, transfer knowledge, and increase standardization.

We asked developers what they see as the main benefits. “Improved Software Quality” and “Sharing Knowledge Across the Team” have remained the top two benefits respectively since 2015.

?

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



n = 1129

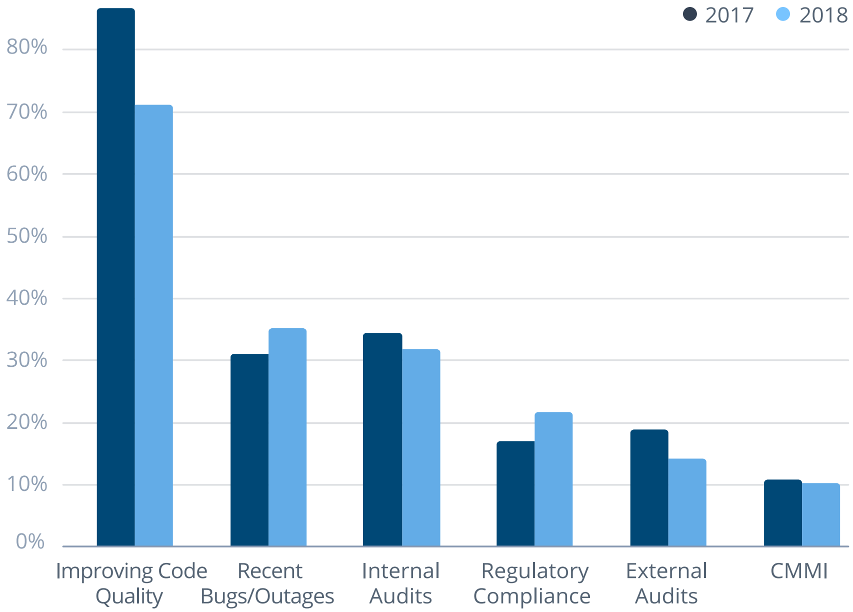
Compared to the last two years, these are the responses that have changed most:

Benefit of Code Review	2018	2017	2016	Net Change
Ability to Comply with Regulatory Standards	31%	24%	19%	**+12%**
Reduced Project Time/ Costs	37%	33%	26%	**+11%**
Adherence to Coding Standards/Conventions	59%	52%	49%	**+10%**
Ability to Set Expectations	28%	22%	18%	**+10%**
Internal Audits	28%	22%	20%	**+8%**
ISO/Industry Certifications	16%	13%	10%	**+6%**

There were no downward trends, which indicates that developers are just increasingly recognizing additional benefits of code review. The upward trend of using code reviews to achieve regulatory compliance was also apparent when we asked the following:

?

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



n = 557 **n = 1107**

As software complexity continues to rise, it is not surprising that regulatory compliance is a bigger priority. With evolving ISO standards and government regulations, development teams have to prove that sufficient software assurance measures are in place. Code reviews offer teams a documentation event whenever changes are made that could potentially result in added defects. Whether for internal or external audits, recorded peer reviews are a useful way to build a software audit trail.

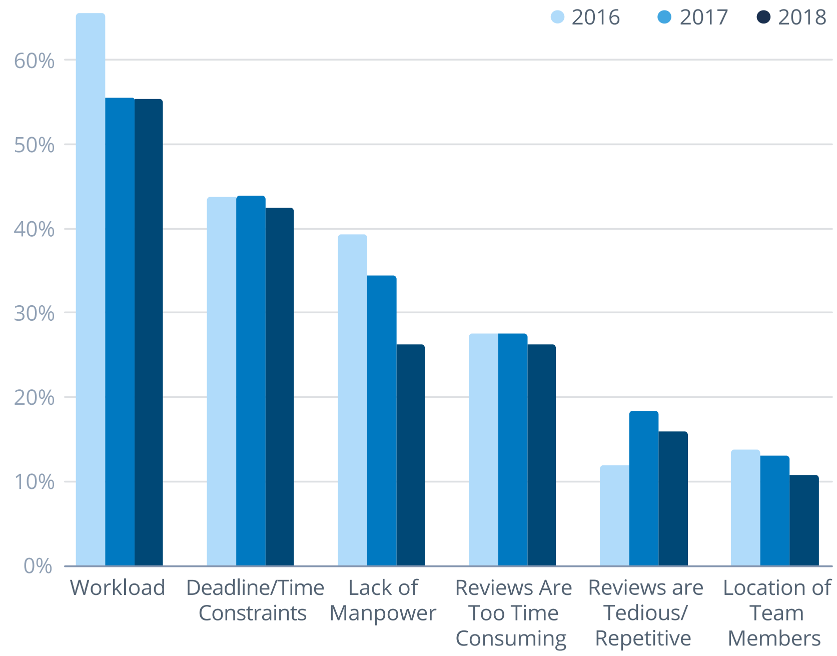
On average, each respondent chose two of the reasons listed. The overall decrease in 2018 percentages in this chart is due to this year's addition of "Not Applicable" as an option for teams that don't conduct tool-based reviews, which garnered 12% of responses.

The increase in "Recent Bugs/Outages" from 31% in 2017 to 35% in 2018 is understandable. This past year has seen record number of software-caused recalls across industries like automotive, healthcare, and consumer devices. Additionally, new cybersecurity challenges are forcing teams to be more careful than ever about preventing bugs and vulnerabilities from reaching production.

Obstacles that Prevent Review Participation

Even though most developers recognize the value of code reviews, there are obstacles that commonly prevent these reviews from happening, or at least delay them. This chart shows the responses from 2016-2018.

What obstacles prevent you from doing any type of code review? Select all that apply.

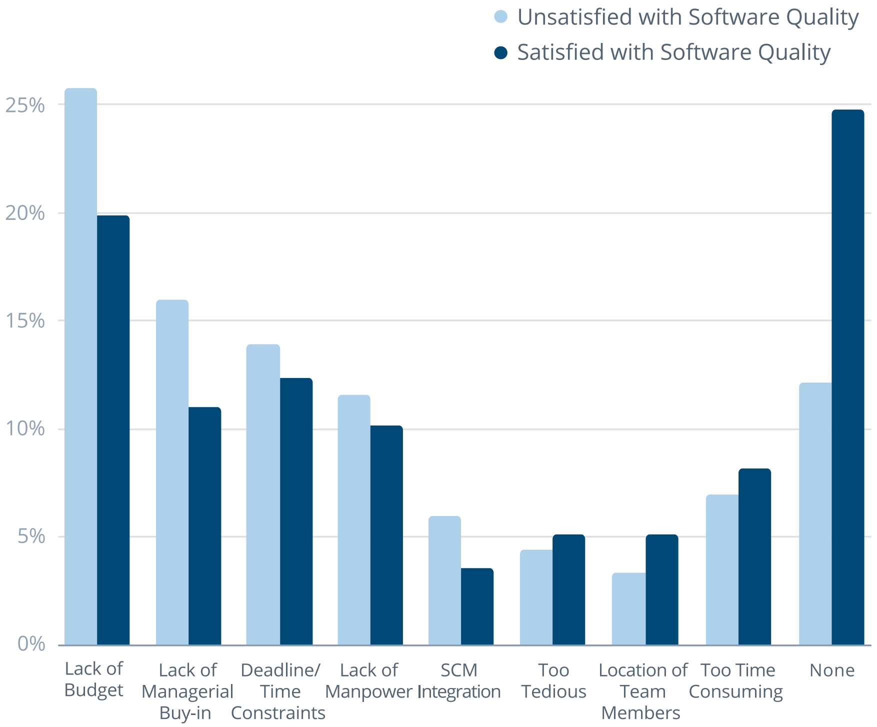


n = 508 n = 511 n = 942

The response that has changed the most over the last three years is "Lack of Manpower". This obstacle has been on the decline with 39% in 2016, 34% in 2017 and now 26% in 2018. This is likely due to the changing size of teams and the distribution of those teams, which we will look at in depth in section #4.

We also asked developers specifically how the obstacles change when the review type is tool-based. This chart below is segmented by the two code review satisfaction cohorts we have been looking at to draw insights.

What obstacles, if any, keep you from doing tool-assisted code reviews?



n = 590 n = 388

The top difference between the cohorts is displayed by the "None" option. 25% of respondents who are satisfied with their code review process say that there are no obstacles to conducting tool-assisted code reviews. Only 12% of the unsatisfied code review cohort chose that option.

Likely related, the unsatisfied cohort selected "Lack of Budget" and "Lack of Managerial Buy-In" as higher obstacles than the satisfied code review cohort by 6% and 5% respectively. This could signal that obtaining a code review tool could be more of a challenge, and as a result might lead to a less satisfactory review process.

Understanding Code Review Expectations

We added a couple of questions this year to get a better sense of how teams are setting expectations and tracking their success. As more teams move towards agile development approaches, the discipline and dialogue around these practices is also growing.

Expectations

Setting expectations can take many forms. It can be as informal as coming to a consensus on the definition of a term or as formal as a using custom checklist with time stamps. Our question this year focused solely on whether developers felt they knew what was expected of them in reviews.

? I know what I am looking for supposed to be looking for when I review a teammate's code.

73% of respondents said that they either agreed **(56%)** or strongly agreed **(17%)**

Only **8%** of respondents said that they either disagreed **(5%)** or strongly disagreed **(3%)**

This is good news for development managers. An overwhelming majority of developers feel that they are comfortable entering code reviews and getting to work with a good sense of their objective.

Reporting and Metrics

The next question is whether those managers and those teams are tracking review metrics and pulling reports regularly.

? My team regularly pulls reports and metrics on our code review process.

Only **30%** of respondents said that they agreed **(25%)** or strongly agreed **(5%)**

40% of respondents disagreed **(22%)** or strongly disagreed **(18%)**

More teams are not tracking metrics and looking at reports than teams that are. Despite iteration and process-consciousness being at the core of the growing agile development movement, code reviews do not appear to be an area of focus. This could be a result of the decentralized review structure used by teams that conduct reviews solely through pull requests in their source control management tool, whether that is GitHub, Bitbucket, or GitLab. The one-to-one nature of these reviews can make pulling reports a much more time-intensive, manual practice. The tools that a development team utilizes have a significant impact on the behaviors of that team. In this next section, we will map the current tools and systems landscape across software development.

Section 3: The Development Stack in 2018

Within the same company and sometimes within the same team, developers are utilizing a diverse set of tools to do everything from project management to bug tracking to requirements management. In this section, we asked developers about the tools and systems that their teams are using.

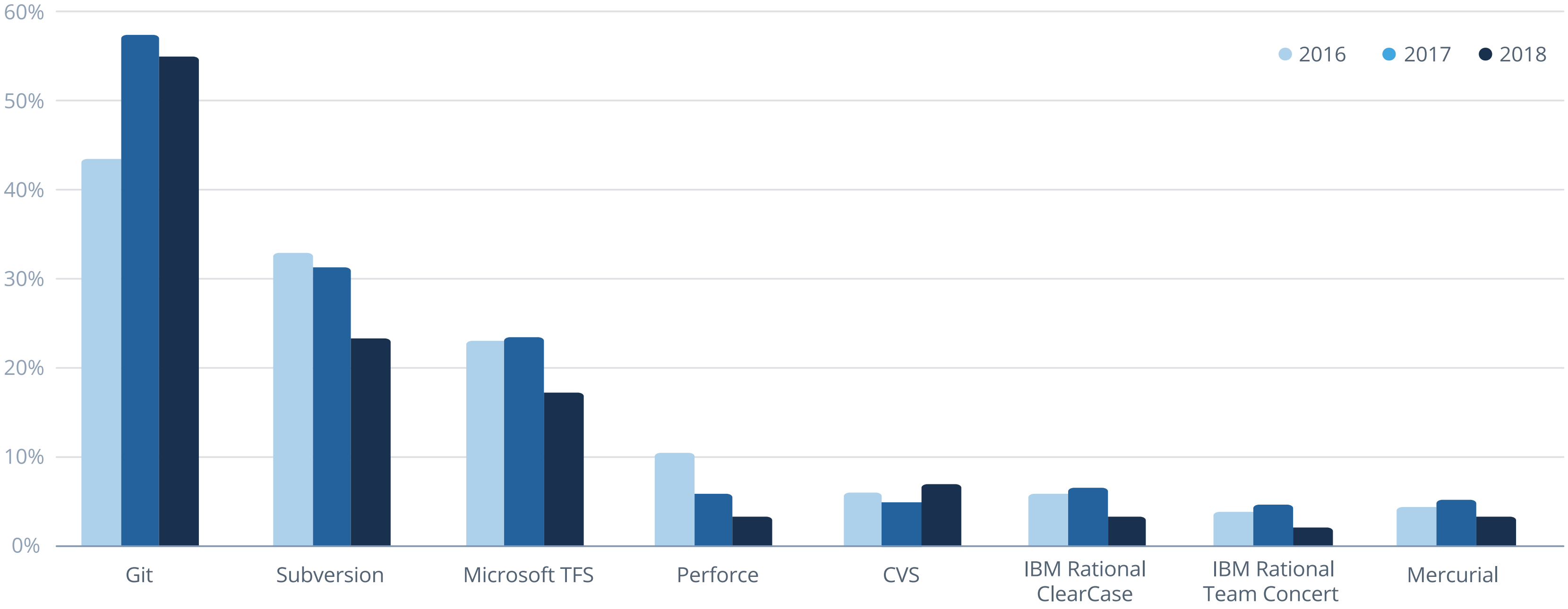
Highlights

- Compared to 2017 and 2016, teams are still overwhelmingly using the same tools, with the only exception being an increased adoption of repository management tools.
- Bug tracking tools are the most common of any category, with 94% of respondents listing a tool that their team uses.
- 42% of respondents are using GitHub, Bitbucket, or GitLab as part of their code review process.

Software Configuration Management Systems (SCMs)

?

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



n = 493 n = 546 n = 908

Git continues to be the most commonly-used SCM system for teams, used by 55% of this year's respondents. Subversion remains the second most common system with 23%, dropping dramatically from 2017 (31%) and 2016 (33%). Microsoft's Team Foundation Server received the third most responses at 17%, down from 23% in 2017 and 2016.

This year, we added "None" as an option. It was selected by 11% of respondents. That accounts for some of the overall percentage drop from 2017 to 2018. The degree to which each system dropped is still indicative of market trends.

We also asked developers about what repository management tool they were using.

- **54% of respondents are using GitHub (44%) or GitHub Enterprise (10%)**
- **30% of respondents are using no repository management tool**
- **28% of respondents are using Bitbucket (14%) or Bitbucket Server (14%)**
- **18% of respondents are using GitLab (14%) or GitLab Enterprise (4%)**

On average, developers who are using a repository management tool selected 1.4 of the options mentioned above. There has been a seismic shift within the last few years around the adoption of these types of tools. In our 2016 survey, 60% of respondents said that they were not using any of these tools. In 2017, that number fell to 34% and now in 2018, that figure has hit a new low of 30%.

GitLab appears to be the fastest growing tool in this area. In 2016, its offerings were used by 4% of respondents. In 2017, that figure climbed to 12%. In 2018, it has reached 18% market share.

Integrated Development Environments (IDEs)

When we asked developers about what IDE they were using, the results looked very similar to previous years.

1. **Visual Studio is the most popular IDE, used by 45% of respondents.**
2. **Eclipse remains the close second, used by 42% of respondents.**
3. **IntelliJ came in distant third with 22% of respondents.**
4. **Netbeans is used by 11% of respondents.**
5. **XCode is used by 9% of respondents.**

None of these tools changed market share significantly compared to our 2017 report.

General IDE usage is on the rise. Only 7% of this year's respondents said that they were not using an IDE, which is down from 8% in 2017 and 13% in 2016. Our data also shows that the IDE market has a number of tools with much smaller user bases. 16% of respondents said that they were using an IDE not listed above.

Bug Tracking

Last year when we asked about bug tracking, 36% of respondents said that they were using a bug tracking tool not listed. This year, we included additional options to try to better capture the most common tools. Despite adding 9 additional choices, we still received 12% of responses for "Other". From that, we can surmise that teams are using a wide variety of tools or workflows to achieve their bug tracking goals.

Atlassian's Jira stands alone as the most prevalent bug tracking tool, used by 55% of respondents.

Microsoft's Team Foundation Server (14%) and Excel (13%) were the only other tools to crack double digits.

The next tier of solutions include: HP ALM (8%), Trello (8%), BugZilla (7%), Redmine (5%), Mantis (4%), Rally (3%), Rational ClearQuest (3%), Rational Team Concert (2%), YouTrack (1%), and FogBugz (1%)

Only 6% of respondents said that they are not using a bug tracking tool at all. That makes bug tracking tools more commonplace than any other tool category.

Requirements Management

Teams are currently managing requirements with many of the same tools used for bug tracking. Each respondent on average selected 1.5 tools from the options we provided.

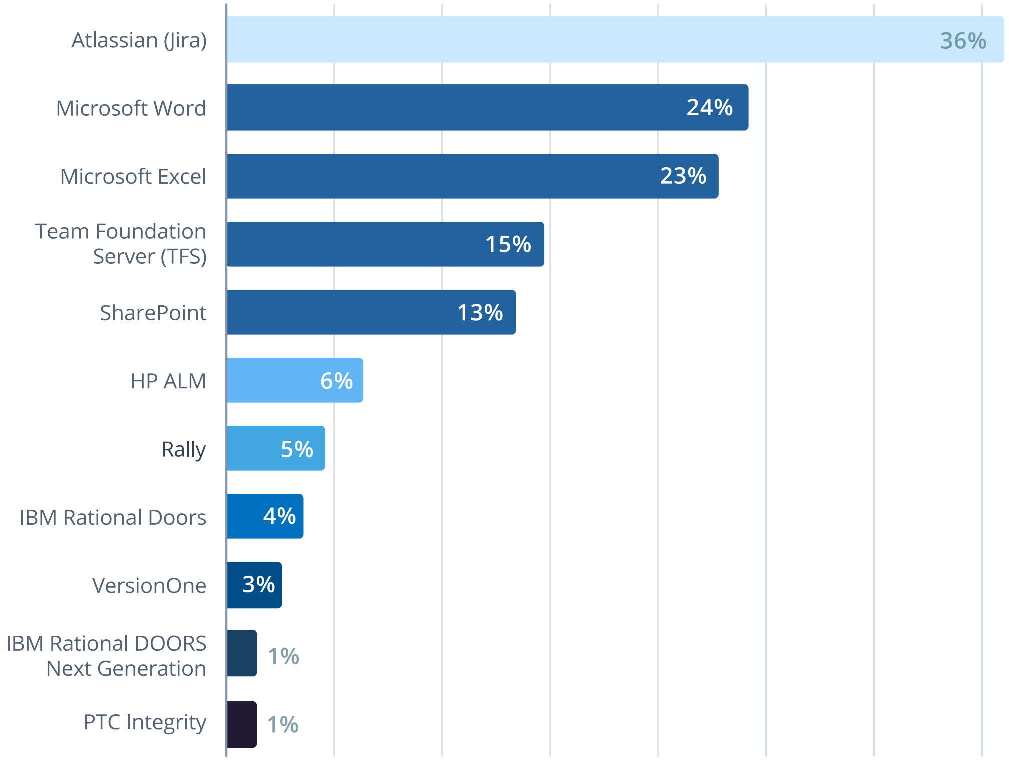
Atlassian's Jira is still at the top of the list, with 36% of respondents using it for their requirements management. The next four most prominent tools are all from Microsoft, with Word (24%), Excel (23%), Team Foundation Server (15%), and SharePoint (13%).

HP Application Lifecycle Management (ALM), which is used by 8.3% of respondents for bug tracking purposes, is also used by 6% for their requirements management. 5% of respondents said that they were using an IBM tool for bug tracking tools, and similarly 5% are using IBM's Rational DOORS or Rational DOORS Next Generation for their requirements management market.

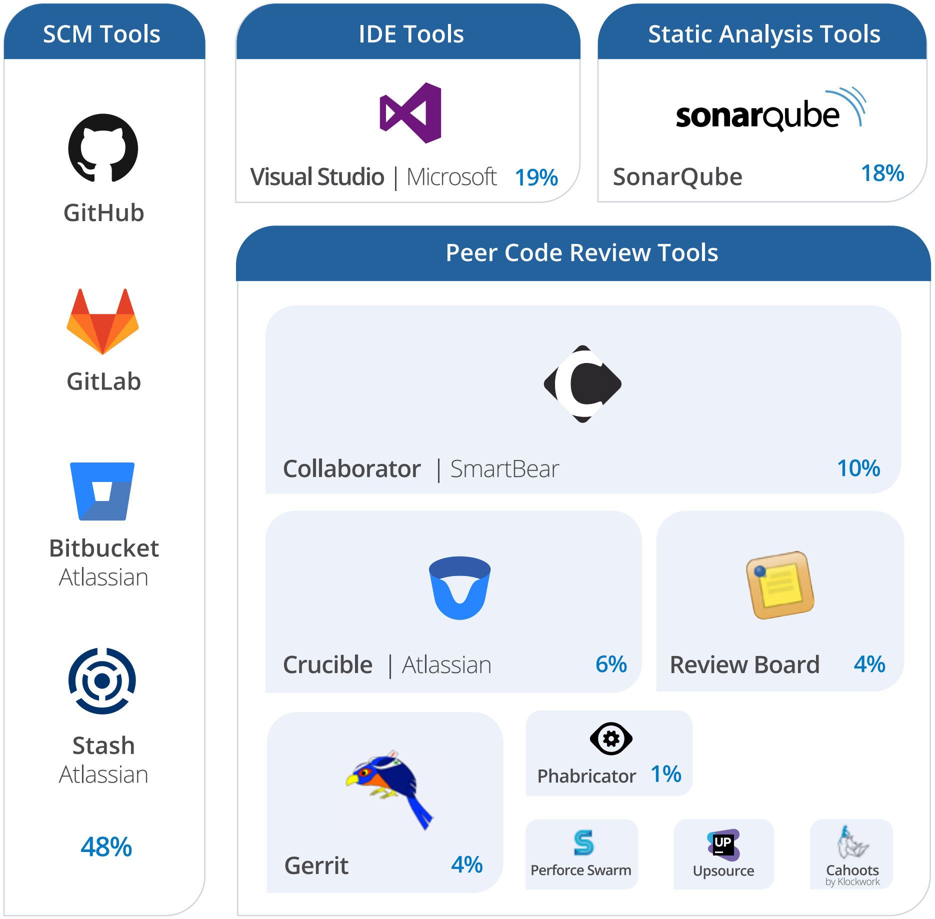
Atlassian leads both the bug tracking and requirements management categories with one tool, but Microsoft ultimately has a much higher share of the market. Each respondent on average selected 1.5 tools from the options we provided. Microsoft tools comprised 48% of all the total responses for this question, while Jira made up only 23% of responses.

?

What tool are you using for requirements management?



n = 902



SCM Tools

GitHub

GitLab

Bitbucket
Atlassian

Stash
Atlassian

48%

IDE Tools

Visual Studio | Microsoft 19%

Static Analysis Tools

SonarQube 18%

Peer Code Review Tools

Collaborator | SmartBear 10%

Crucible | Atlassian 6%

Review Board 4%

Gerrit 4%

Phabricator 1%

Perforce Swarm

Upsource

Cahoots
by Klockwork

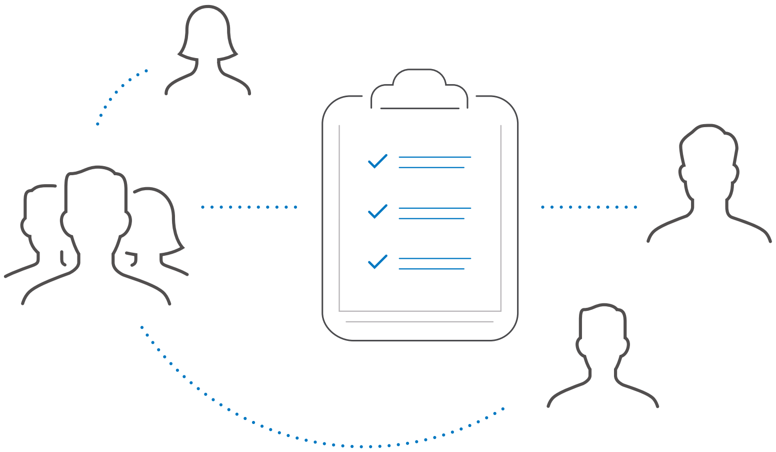
Code Review

Given the primary topic of this report, understanding what tools developers are using to conduct code reviews is critical. There are four segments of tools for this category, repository management tools, IDEs, static analysis tools, and peer code review tools.

- Repository tools like GitHub, GitLab, and Bitbucket enable developers to conduct reviews by sending pull requests to team members.
- IDEs like Visual Studio and Eclipse have varying degrees of review capabilities.
- Static analysis tools like SonarQube and Veracode enable general spot-testing of code.
- Peer code review tools like Collaborator by SmartBear and Crucible by Atlassian empower developers to review each other's code, identify bugs, and provide feedback.

Section 4: Team Firmographics and Beliefs

In this section, we will look at the size and disposition of development teams across industries and how they have changed over the years. We also asked a few questions aimed at gauging how open teams are to new trends in software development.



Highlights

- Development teams of 6-10 people are becoming more common year over year, representing 38% of teams in 2018.
- While many developers are talking about no code/ low-code development, only 18% of teams are considering moving towards that approach.
- Teams are more international than ever, with 40% of respondents saying that their team is distributed across two or more countries.

Trends and Beliefs

We included two questions to gauge how teams feel about trends in development and data storage. Over the last three years, we have tracked the sentiment from respondents on the following:

My team is comfortable with storing source code in a SaaS tool.

- 32% of respondents said that they agreed (25%) or strongly agreed (7%) with this position.
- 27% of respondents disagreed (16%) or strongly disagreed (11%). Strongly disagreed was higher in both 2017 (14%) and 2016 (16%).
- Besides a downward trend among respondents who strongly disagree, there has not been significant change in responses over the last three years.

We asked respondents for the first time this year about no code/low-code development, which has become a much more talked about approach in recent years.

My team is currently utilizing or considering moving to no-code/low-code development.

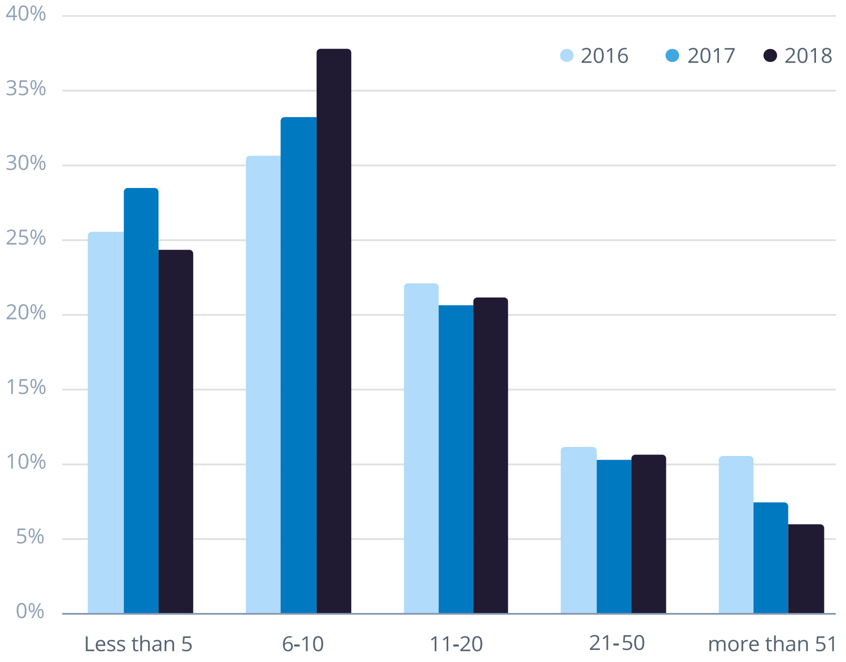
- Only 18% of respondents agreed (15%) or strongly agreed (3%).
- 41% of respondents said that their team was neutral to the topic.
- 45% of respondents disagreed with this statement.

Despite being considered a trend by many, the actual of adoption of no-code/low-code development techniques and tooling seems to be relatively small. Since we only have one year of data, we can't say anything about the rate of adoption. Our 2019 report will show whether this approach is gaining traction quickly or not.

Size of Development Teams and Companies

Earlier in this report, we highlighted that "Lack of Manpower" as an obstacle to code reviews was decreasing year over year. In part, this seems to be the result of changing team sizes. In this chart, you can see the size of development teams over the last three years.

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



n = 854 n = 566 n = 493

Teams with less than 5 people are at a three year low, representing only 24% of respondents. The most notable shift is a move towards teams sized between 6 to 10 individuals, comprising 38% of respondents in 2018. This is up from 33% in 2017 and 31% in 2016.

At the same time, teams of more than 51 people have been steadily declining, from 11% in 2016 to only 6% this year. Teams of that size are likely working on enterprise-grade projects, but the sheer volume would make it difficult to adopt modern development practices like scrums and retrospectives. Even if these large teams are developing in a waterfall framework, the attractiveness of smaller teams and agile tactics may be a catalyst for this shift.

We asked respondents the following to know what data biases might exist in these trends:

You can see in this table that the distribution across these different company sizes is fairly even. As a result, the trends and insights in this report are valid and applicable regardless of your own company's size.

Respondent Firmographics

Roles Surveyed

The first question of our survey this year was the following:

Which best describes your role?

On average, respondents selected 1.5 job titles. This could be a sign of the increasingly cross-functional nature of development teams and trends towards shifting testing left in the software development lifecycle. Overall, developers comprised 59% of the respondents, followed by software architects (23%), testers (18%), and development managers (14%).

?

What is the total number of employees at your company?

Number of Employees	Response Percentage
Less than 25	**14.2%**
26 – 100	**16.3%**
101 – 500	**19.3%**
501 – 1000	**11.7%**
1001 – 10,000	**17.7%**
10,001+	**20.8%**

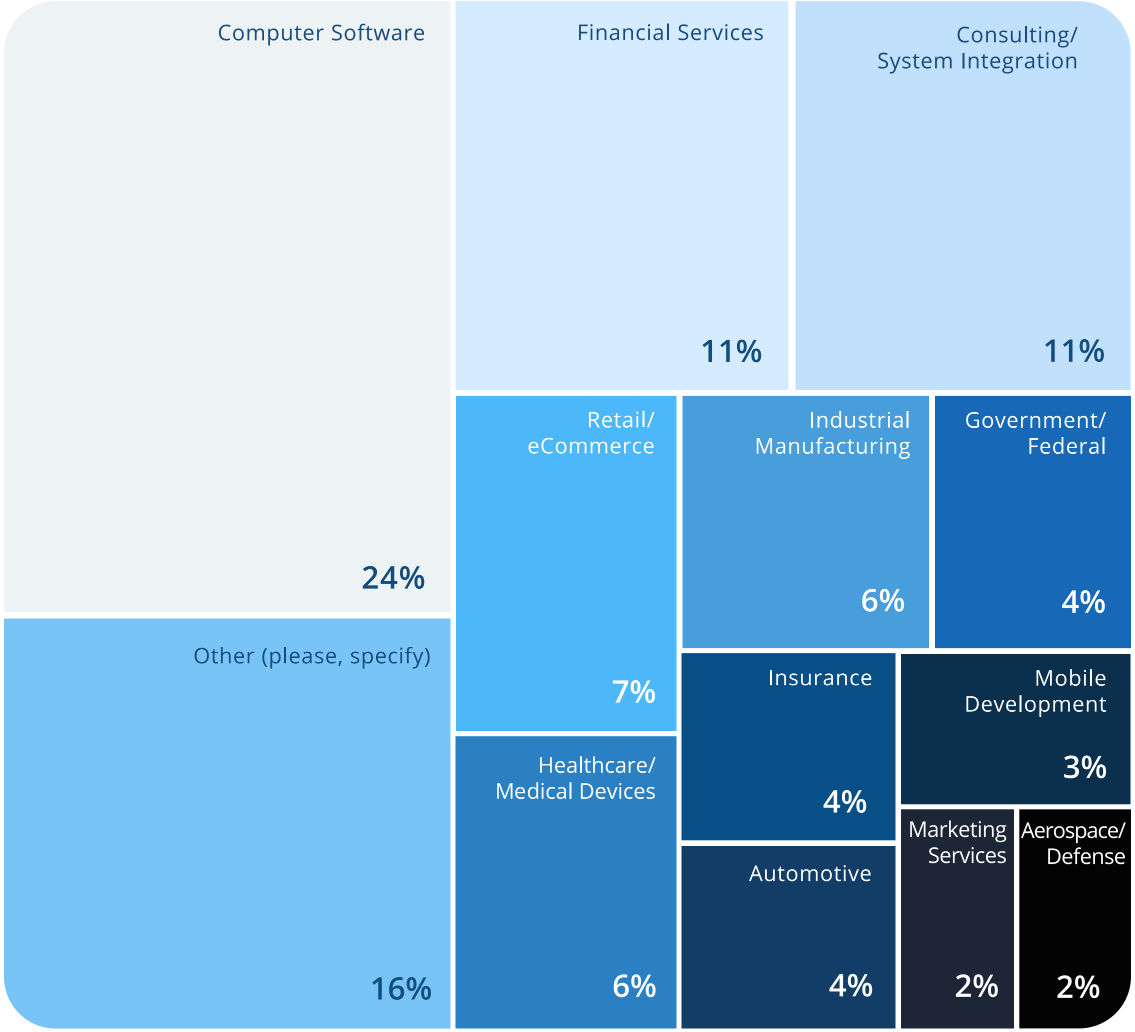
Industries Surveyed

In this year's survey, we heard from individuals working in at least 35 different industries. This is an increase from last year's 30 industries, likely due to our nearly double amount of respondents. This chart shows the respondent breakdown by industry.

Because of our wide audience of respondents, the findings of this report are generalizable across many roles and industries doing software development. In this final section, we will provide actionable insights and recommendations that your team can act on to improve your code review process.

n = 854

? What industry do you work in?



Section 5: Recommendations for Your Team

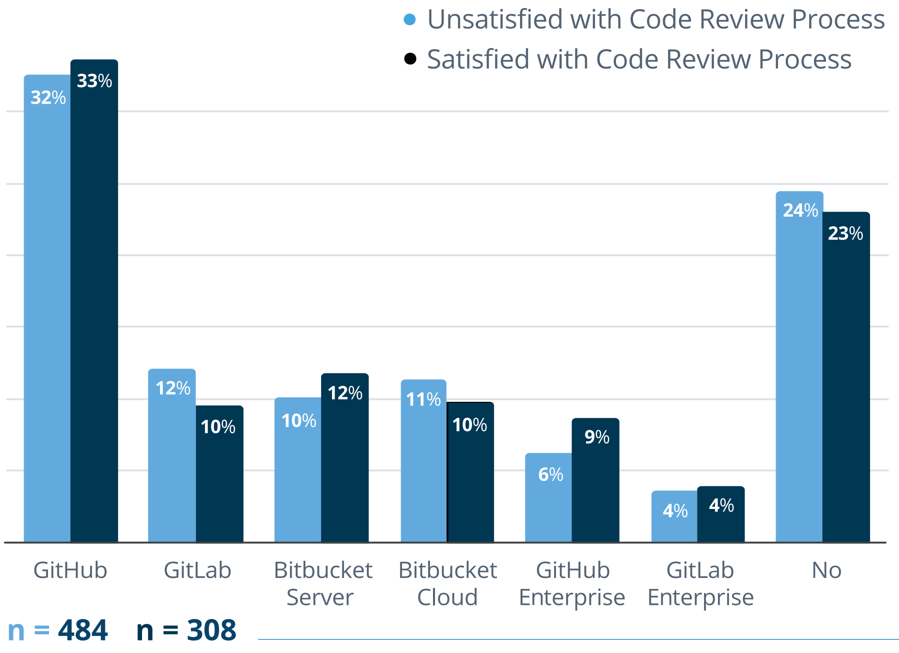
Every development team would like to be satisfied with their code review process. In this section, we isolated the key differences between teams who are dissatisfied or very dissatisfied with their code review process and those who are satisfied or very satisfied. For the sake of clarity, we removed respondents who said that they were neutral when asked if they were satisfied with their code review process.

*These are **7 recommendations** that any team can use to improve its code review process:*

1. Don't worry about what repository management tool your team is using.

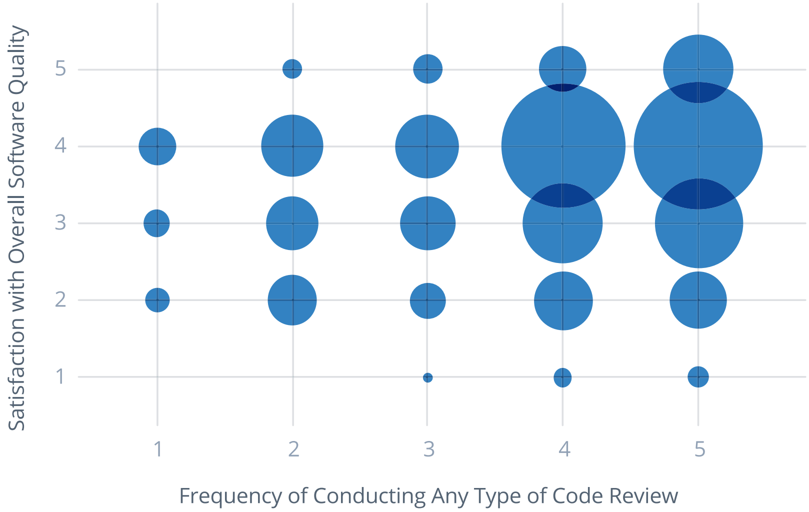
It does not matter what repository management tool your team is using, if any, when it comes to code review satisfaction. 42% of respondents said that one of these tools was part of their code review process, but these tools themselves have no significant impact on whether a team has a satisfactory code review process. If you are looking to improve your code review process, evaluating new repository management tools is not the way to do it.

Are you currently using any of the following repository management tools?



2. Conduct code reviews on a daily basis.

Satisfied teams are twice as likely to perform code reviews on a daily basis (45%) than teams that are dissatisfied (22%). While daily code reviews might sound like a criminal sentence to some developers, high review frequency translates to better code quality. There is a compounding effect when you introduce all the benefits of code review into daily behavior. Communication improves, knowledge about the codebase is shared, and fewer bugs make it through development to QA.



n = 713

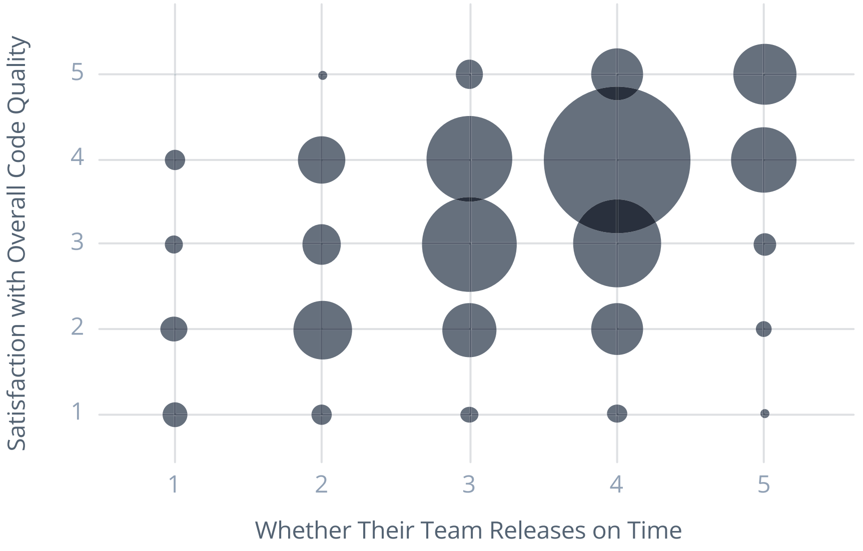
3. Think of code reviews as a time saver, not a time expense.

76% of respondents who are satisfied with their code review process say that they are able to get releases out on time compared to the 44% from the unsatisfied cohort. This chart shows the correlation between respondents who say that their team is able to release software on time and respondents who say that they are satisfied with the overall code quality they help deliver. In short, releasing on time is a strong indicator that your team is satisfied with the quality of your software and vice versa.

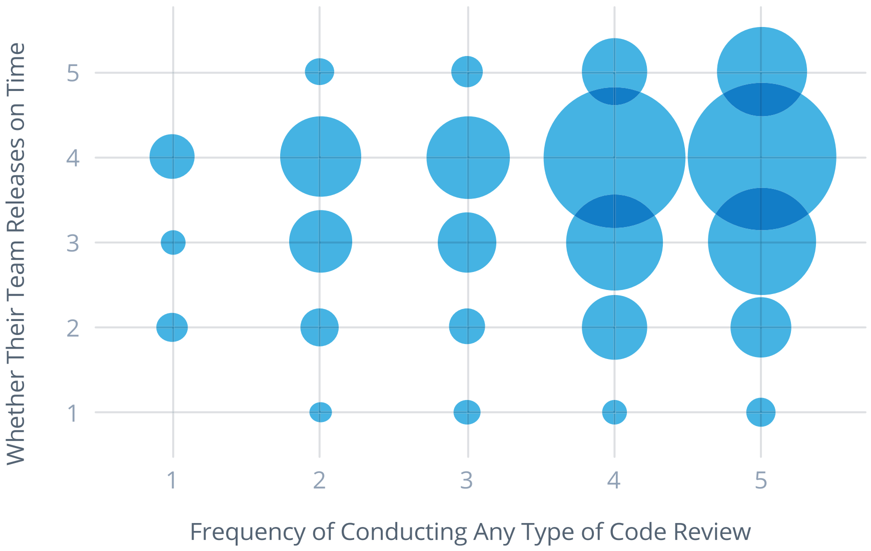
But what if your team is continually missing your release dates? Though code reviews take time, teams who have harnessed their process appropriately are able to ultimately shorten their time to market by reducing bugs early, thus reducing time spent in rework.

This chart below shows that there is a strong correlation between respondents who conduct frequent code reviews and teams who are able to release on time.

There is definitely an investment to get your review process refined and churning, but the impact on development time can be transformational.



n = 715



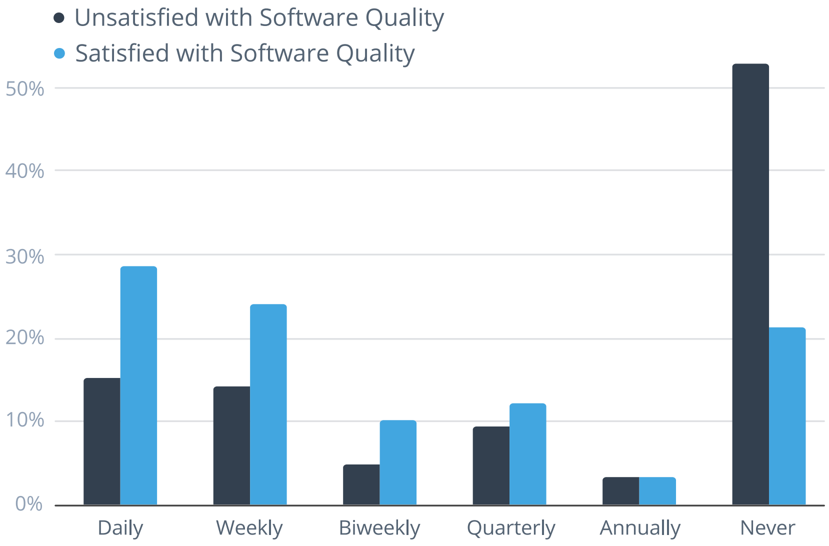
n = 715

4. Conduct tool-based reviews.

Teams that perform tool-based reviews are more likely to be satisfied with their overall code quality. Cohorted by code review satisfaction, 79% of satisfied respondents are conducting some kind of tool-based reviews. Comparatively, only 47% of unsatisfied respondents are conducting tool-based reviews.

If you are not currently, start conducting tool-based code reviews. Your team is more likely to be satisfied with your code review process, and subsequently more satisfied with your overall code. If you are a developer on a team that needs additional budget or managerial approval to make this possible, share this report with them or read our recent blog post on how to quantify the return on investment in your code review process.

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



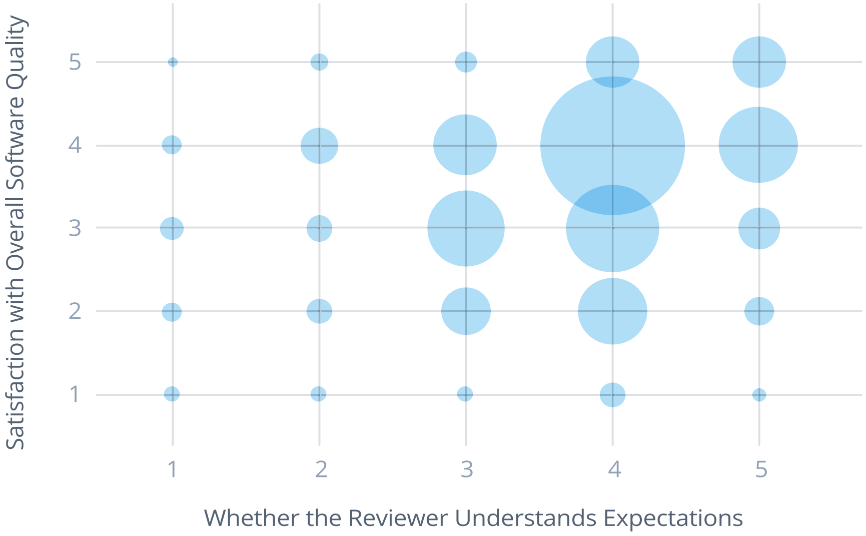
n = 244 **n = 398**

5. Set clear expectations for both code and document reviews.

Earlier in this report, we showed that 73% of respondents say that they know what to look for when they participate in a review. This chart below shows the correlation between respondents who say they know what they are looking for in reviews and respondents who say they are satisfied with their overall code quality.

The more clearly you can set expectations, the more likely it is that your team will produce higher quality software. This applies to both code and document reviews. We mentioned earlier that the satisfied code review cohort performed reviews on more types of documents and more often leveraged a tool for their process than the unsatisfied code review cohort.

How can you ensure that your team is setting clear expectations? Consider outlining expectations explicitly in a checklist. Code review tools like Collaborator allow you to build custom checklists in review templates so participants can easily see what things they need to be looking for on each kind of project.



n = 847

6. Use tool-based reviews in conjunction with meeting-based reviews.

Of the teams that never perform meeting-based code reviews, only 28% are satisfied while 48% are dissatisfied with their code quality. Just because teams that conduct tool-based reviews are more satisfied does not mean that you should only conduct tool-based reviews. Meeting-based code reviews, even if they are remote, provide a meaningful opportunity for your team to define a culture, to share successes, and identify areas to improve.

Additionally, take advantage of meeting-based reviews to provide clarity when possible and clear up miscommunication that can occur through text. Setting clear expectations also mean talking through acceptable timelines responding to reviews or who might be the best participants to add to a certain kind of review.

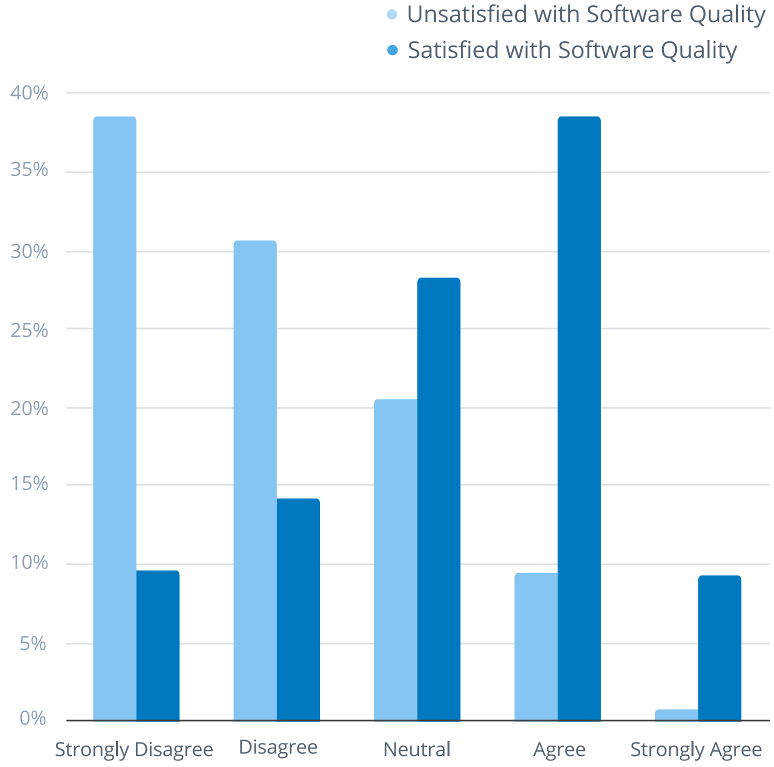
7. Choose review metrics to track and make reporting part of your process.

48% of satisfied reviewers regularly pull reporting on their process, nearly 5 times that of unsatisfied reviewers (10%). It is hard to be satisfied with your code review process when you can't track its aggregate effectiveness. Teams that do, through review reports and key performance indicators, and more likely to know what to improve so they can make changes.

Adopting a tool that enables you to track key metrics and pull custom reports on your peer code reviews is the fastest way to start driving meaningful process improvement.

?

My team regularly pulls reports and metrics on our code review process.



n = 254   n = 407

Fostering a Collaborative Culture

Besides checklists and meetings, ensuring that your team is always comfortable and empowered to communicate with each other is critical. Building a successful review process is about more than just adopting a tool, conducting daily reviews, and tracking metrics. It is about fundamentally empowering team members to own their code.

Just being conscious of how your team is communicating, from language to medium, can allow you to anticipate problems before they manifest. A culture of constructive collaboration can be the powerful undercurrent that propels your team's development to the next level, as long as you are willing to make the investment in your process and commit.

SMARTBEAR

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

6.5M+	**22K+**	**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