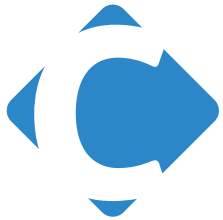

A Guide to Improving Code Quality
& Collaboration
on Your Development Team
Collaborator



# Collaborator

Collaborator is a code review tool that **helps development, testing and management teams work** together to produce high quality code.

LEARN MORE ABOUT **COLLABORATOR**

# Content

**Earlier this year, SmartBear Software released our *State of Code Quality 2016* report.** The report included input from more than 600 software professionals, across more than 30 industries.

Participants in the survey work on software teams ranging from less than five employees up to more than 50 employees, and work for companies ranging from small businesses, with less than 25 employees, to enterprise organizations, with 10,000 employees or more.

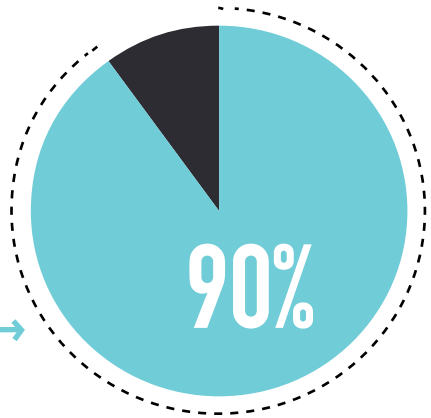
The goal of the report was to determine how software teams are building and maintaining code quality in a world where teams are being asked to move faster than ever before.

When asked for the number one thing they felt organizations could be doing to improve code quality, the #1 response was code review.

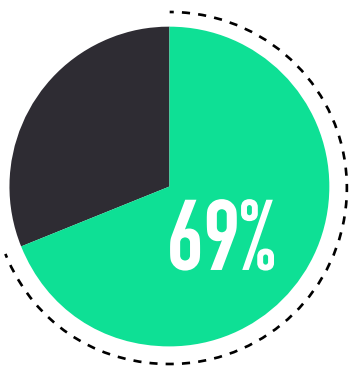
Furthermore, when asked specific questions about the benefits of code review and the factors they consider when choosing a code review tool:

**90%** of respondents say that improving code quality is the biggest benefit of code review

code review =
code quality →




**69%** of respondents say that improving code quality is the biggest business driver determining the need for a code quality tool




code quality =
business driver

Organizations – from small businesses, with less than 25 employees to enterprise organizations with more than 10,000 employees – agree that when it comes to improving software quality, code review is king.

While code review is a critical component to code quality; it is just one piece of the quality puzzle. Reviews may be able to catch defects before code is sent to QA, but if there aren't standards for the code you are writing and there isn't a clear understanding of the user requirements, you will continue to run into bugs that can be costly for your organization.

In an increasingly agile world, collaboration on development teams needs to go beyond peer code review. Software quality is driven by collaboration throughout the entire development lifecycle. The move from waterfall to agile development has meant faster paced development efforts, with a need to shorten feedback loops. This requires tools that support the process. These tools include integrated development environments, build and test automation tools, requirements management and bug tracking tools, as well as peer review tools.

How can your development team work together to reduce defects, improve code quality, and bring these tools together?

This eBook will discuss the idea of code quality and how it fits into dev collaboration.

## We will cover:

- Defining code quality
- Understanding the role of collaboration in improving code quality
- How a collaborative approach to code reviews will help improve code quality
- Overcoming the obstacles to peer reviews

# Defining Code Quality

**What one developer values in terms of code quality may be very different than what another developer values.**

A simple example of this is a method written very compactly, using concise syntax and as few lines of code as possible. While this may be seen as high quality code by one developer, it may be seen as terse and esoteric by another developer who prefers to have code laid out in a more verbose and easy to read structure.

While we may never be able to agree on a singular answer to the question, "What is code quality?" There are certain factors that should be consider when determining the quality of your code.

## Establishing Coding Standards

No matter what programming methodology your team has adopted (waterfall, scrum, test driven, extreme, or Agile) maintaining a standard for your code base is vital to project momentum. It will also help address the personal biases that can come into play when attempting to defining what "good code" looks like for your organization.

Establishing code standards, either through a doc or adding a coding formatter style sheet to a preferred IDE removes a couple issues within the development process, including improving version control and taking ego out of the mix. Removing coding styles makes the end-result (your code base) seamless and alleviates the concern that one developer is changing another developer's code.

While using some sort of style sheet to compare against provides clarity and ownership as a group to the entire team. It also protects against the potential for having to clean up after a code import that causes widespread issues in readability and functionality.

## Meeting the Requirements

Another measure of quality to consider is simply whether or not the code meets the requirements that caused it to be written. Under this definition, code expressed beautifully in the eyes of a developer, but failing to accomplish the intended outcomes, can be said to have low quality. The resulting software isn't realizing the most basic measure of fitness: it must actually meets the customer's needs.

Beyond customer requirements, there is also a varying set of other requirements that need to be considered, including: internal requirements, marketing requirements, performance requirements, and regulatory requirements for specific industries.

Of course, software can meet requirements and still have poor quality. Satisfying customer's requirements may be required for high code quality, but there is more to quality than this single attribute.

Focusing on only this minimum level of quality begins to cause problems as software needs to change to meet new requirements. Today, software typically grows as new customer requirements and desires are added and expressed into the code. Therefore, code must also exhibit the quality of being easy to change.

## Maintainability

When you build software, you stack the new atop the old and everything comes along for the ride. Sure, there may be the occasional new module that stands all by itself or plugs in with the rest, but that's the exception. The new code interacts with the old stuff, calling into it, relying on it, and running beside it in production.

The fact that someone typed it out years ago doesn't mean that it doesn't have a very real, current impact on your team every time you deploy your code. Because of this, code needs to be able to be changed and maintained over time. If software is difficult to change, it is considered to be high in technical debt. A code base with high technical debt has a higher total cost of ownership because changes take longer to implement and the risk of introducing new defects while adding new features is high.

## Free of Deficiencies

Deficiencies in a software context are a very large concept, because there are so many aspects of code that can be observed, measured, quantified, and qualitatively assessed.

We can take "freedom from deficiencies" as referring to well-performing non-functional requirements such as maintainability, readability, and scalability. Since there are so many qualities of code to consider, it quickly becomes a matter of deciding which aspects to optimize, and which to monitor less aggressively.

Identifying these deficiencies will involve a number of other artifacts that will need to be tracked throughout the development lifecycle, including: design specifications, release/sprint backlog items, code reviews, test cases, bug reports, and product documentation.

## Reducing Time-to-Context

The amount of time a developer must spend to get enough context to make can be referred to as time-to-context. Creating maintainable code is done in the pursuit of reducing time-to-context for the code reader. Things that promise reductions in time-to-context include clean separation of concerns, ensuring each class has a single responsibility, and simple designs.

Maintaining low technical debt and a fast time-to-context for the reader go hand-in-hand in ensuring that code is easily maintained. Further, the simpler code can be, the less likely it will be to contain defects. Thus, maintainability goes a long way to meeting the second requirement that code be free of deficiencies.

## The Reality of Code Quality

As part of *The State of Code Quality 2016* report, we asked developers to rate their satisfaction with the quality of software they help build. Overall, we found that respondents felt positively about the quality of the software they help build. Of those who didn't either agree or strongly agree, roughly 10% said they felt negatively about the software they help develop. The overall positive sentiment was felt across industries and roles within software teams.

But while the overall sentiment was positive – with two-thirds of respondents saying that they are satisfied with the quality of software they help build – there is still room for improvement. This is especially true on larger development teams. One in five respondents on development teams with 50 employees or more said they either disagreed or strongly disagreed that they were satisfied with the quality of software they help build.

As outlined at the start of this eBook, achieving and maintaining a high standard of quality depends on collaboration. But what does collaboration really mean in a modern development team?

# Collaboration in Pursuit of Quality

At each phase of the development process, collaboration is critical. You may already be doing some form of collaborative software development, but improving the efficiency of this process brings added speed, quality, and agility. Developer collaboration encompasses the following: idea creation, user stories, product requirements, source code, test artifacts, and ultimately deployment.

In the idea generation phase, a good room and a white board might be all that is needed. But as user stories and requirements are written, the ability to provide feedback and track changes becomes even more important. For some regulated industries, development procedures require audit trails that can be quickly provided to the governing bodies. Documenting the conversations around requirements is just as critical as other later stages of product development.  When team members leave and future teams need to review and make changes to existing legacy code, they will be able to refer to these documents and understand why certain decisions were made and changes took place.

## For all companies writing software, collaboration should be a top priority.

In many cases, that starts with establishing a documented code review process. A code review process ensures that teams and organizations are releasing high quality software downstream to test engineers and ultimately customers. The tools needed for code review vary from team to team as well as organization to organization, but most times it is driven by a team's desire to release code quickly with as few defects as possible.

Most organizations also pursue distributed development. These distributed global teams require a way to easily provide feedback on source code with shortened feedback loops due to time zone challenges.

According to the *State of Code Quality 2016* report, more than half of development teams are working across multiple locations. Thirty-eight percent of respondents are working on development teams located across two or more countries.

Without a robust developer collaboration tool, collaborating across time zones can be tedious and time consuming, and can cause significant loss in development velocity.

## Once the code is written, collaboration doesn't end.

In fact, testing and the release process require collaboration. Though many organizations automate portions of their QA tests and deployment scripts, artifacts in both phases still need to be reviewed.

**For QA, these reviewable artifacts might include:**

- Test plans
- Test cases
- User scenarios
- Requirements

Ensuring that the entire team knows what a successful test process looks like, helps maintain quality before deployment. When it comes to deployment teams, many forget to review their artifacts. Reviewing infrastructure code is a critical step to verifying that deployment on release day involves as little down time as possible. This has never been more true than it is today with the move to DevOps.

Due to the rapid pace of development and deployment, individuals often see this collaborative approach as time consuming and make the argument that their workload is too heavy to participate.

Yet, it has been proven to save time and money and lighten workload. This is due to the fact that defects are easier to find earlier in the process. As you get further from user stories/requirements and more into the QA process, defects become increasingly more difficult to find and more expensive to fix.

**Done correctly with the right tools, at the right part of the lifecycle, developer collaboration can:**

- Help bring new team members up to speed faster
- Find defects earlier – before they ship to customers – cheaper to fix early
- Create audit trails
- Minimize risk due to turnover
- Knowledge share – more than one individual knows what is going on
- Distributed teams communicating daily
- Faster onboarding

## Why code review is an essential part of dev collaboration.

This collaborative approach to software development often begins with peer code reviews. Why? On a team that does code reviews, everybody understands the codebase and its data flow.

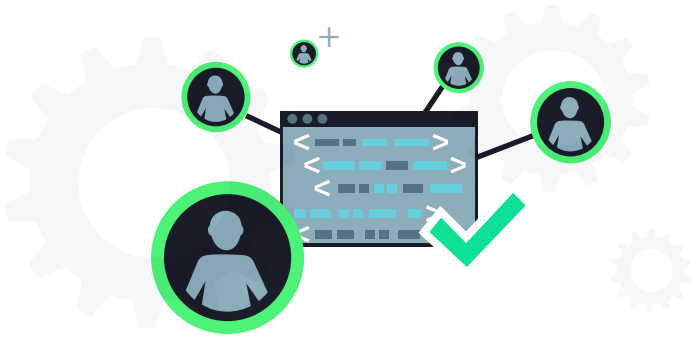
More importantly: everyone understands the reasoning behind the decisions that were made when changing the code, not just the mechanics of the code. For instance, if everyone recognizes that this application needs to be lightning fast, the code written (and reviewed) will be created with that priority in mind. That makes it easier to make choices across the project, since everybody is on the same page for the "why."

A person uninvolved in the project should be able to read a portion of code and understand what it does, see the constraints and preconditions of its use and contextual information (e.g. the author used a peculiar technique because of a bug in the OS). In addition, software organizations often have coding guidelines ranging from whitespace conventions to the maximum size of a function. A reviewer can provide the ignorance and objectivity necessary to ensure these goals.

Code reviews help create internal standards. The end result is that a new team member is exposed to the organization's methods (programming and otherwise). By the time the new developers need to modify code, they understand globally what is going on.

It shouldn't come as a surprise that, in addition to improving code quality, participants in the *State of Code Quality 2016* report also cited, sharing knowledge across the team, ability to mentor less experienced developers, and increased collaboration as benefits of code review.

Your team may already be doing periodic reviews of code but haven't invested in setting up a healthy peer review process for your development team. Luckily, implementing a code review process doesn't require a major overhaul of current processes and won't need to disrupt the effectiveness of your development team.



# How Code Reviews Help Improve Code Quality

Part of the joy of creating software is in the individuality each of us brings to the problem of expressing intent as code. As a developer, your challenge is in ensuring that expressions of intent as code are easily understood by others, while meeting the quality standards of delighting the customer.

Authors need editors to catch their mistakes. It is human nature that one cannot adequately proof-read one's own work. Authors of software need the same assistance as authors of novels to achieve the goals of the software organization.

The purpose for code reviews are as diverse as the environments in which they are conducted. However, almost all code reviews have these goals in common:

- Defect-free, well-documented software
- Software that complies with enterprise coding standards
- Teaching and sharing knowledge between developers

Other objectives often include: maintainability, security, consistent end-user documentation, adequate comments in code, complete unit tests, and scalability.

## How code review works

A code review involves one or more developers examining source code they didn't write and providing feedback to the authors, both negative and positive. Ideally, even when the reviewers are not as involved with the project they are reviewing, it maximizes objectivity and ensures the code is readable as well as maintainable even by those not already well-versed in that project. Typically the reviewers will have a standard checklist as a guide for effectively performing code reviews so that finding defects is top of mind and to validate the code against the company's coding standards.

## When should code reviews take place?

Code review takes place during all stages of development, with the exception of small projects such as demos and experiments that are designed to be written quickly and most likely not used in production. Even during the final stages of development when everyone

is trying to meet a deadline, code review greatly reduces the number of regression bugs and ensures that company coding practices are not abandoned.

## Improving code quality with code review

We all want to produce higher quality code so that, when we ship the newest release, we know the code is solid for your customers. As every developer has learned painfully, just because something in your code "works, doesn't mean it's actually done right."

It is known that code reviews help developers find bugs sooner, discover user-story assumptions before they're instantiated into code, and fix problems that would pass automated tests. The earlier in the software development cycle that a problem can be spotted, the cheaper it is to fix – in time, dollars, and frustration.

**Among the types of bugs found during code reviews:**

- Missed test cases: "You tested for an empty string, but not for null."
- Typos
- Un-optimized code
- Unnecessarily complex code
- Re-implemented code: "We already have a function that does what you're doing here, and does it better."
- Lazy documentation

# Overcoming the Obstacles to Code Review

While participants in the *State of Software Quality 2016* report understood that code review was the #1 things they could be doing to improve code quality, they also acknowledged challenges that stand in their way.

When asked, what obstacles prevent you from doing the level of code review that you desire?

- **66%** said that workload is the biggest obstacle
- **44%** said that deadlines/time constraints are the biggest obstacle
- **39%** said that lack of staffing is the biggest obstacle
- **28%** said that reviews are too time consuming

One of things we commonly see happen on teams that are attempting to manage workload while still maintaining software quality is to adopt an ad-hoc, or "over the shoulder" approach to code review.

While this approach to code review can be useful on certain projects, it also has the potential to become a blocker in your development process. In fact, when we asked participants who were doing ad-hoc code review how frequently they were doing it, less than 5% said they doing code review daily. On the contrary, 23% of teams that were using a tool for code review were able to do it daily.

"66% said that workload is the biggest obstacle"

There are a number of benefits of using a tool to assist with code reviews. Here is a quick overview of the key tools and functionality that come with doing tool assisted code reviews:

1. Central location to review and discuss user stories, requirements, code and test plans
2. Comment and defect creation and the ability to verify that the appropriate changes have been made
3. Customizable workflows can be configured for each team in the enterprise
4. Ability to Review Word, PDF, Images, and text files (source code) in their own unique workflows
5. Reporting and audit trails for regulatory & standards compliance
6. Checklists that ensure key items are being checked during the review
7. Electronic signatures that verify review signoff
8. Automatic linking to bug tracking utilities for easy reference to issues
9. Configurable roles and responsibilities for the review to ensure the right people are participating
10. Ability to run scripts based on events in the review to trigger actions in 3rd party utilities
11. Integrations different SCMs that make attaching files to reviews quick and easy

# Collaboration is Action

We're constantly speaking with users of SmartBear's Collaborator product to find out how they are using a dev collaboration tool to improve code quality. Each had business challenges they wanted to address when looking for a tool and the benefits of each organizations has gone beyond the quality of software they deliver.

## Everi Games — Streamlines Code Reviews for Software Operating Electronic Gaming Units

ColEveri's mission is to be a transformative force to casino operations by delivering reliable protection and security, facilitating memorable player experiences and striving for customer satisfaction and operational excellence. The Everi Games software team consists of approximately 80 developers and QA testers who work from the company's gaming headquarters in Austin as well as gaming studios in Chicago and Reno.

Everi uses Collaborator to peer review code — as well as user stories and test plans — in a transparent, collaborative framework while also keeping the team up to speed in real time on code changes. In addition to improving the quality of the software his team helps build, Tim Moore, technical director for Everi, says that using a tool like Collaborator has helped to breakdown silos, as well:

*"Our gaming systems are generally healthier because each team can take a close look at the code during the design phase, and more people understanding the moving parts makes each product holistically superior... Collaborator also enables us to break down knowledge silos — our intellectual property is not isolated on each developer's desktop."*

The tool has also helped identify problem areas and better manage the team's workload.

*"In cases where we see a spike in defects from an experienced developer, Collaborator helps us analyze whether we might be rushing development, or perhaps the requirements are nebulous so that the developer is struggling to fulfill them. The issue might also be due to two groups that may not have a good feel for each other's style and syntax. With Collaborator, we can more easily identify the specific cause."*

Key takeaway: As a company that provides electronic games to casinos and race tracks, Everi Games must ensure its software performs reliably and conforms to gaming industry regulations. To address this challenge, Everi Games deployed Collaborator as its peer code review tool. Developers and QA testers can now work together more efficiently, and their managers can more effectively oversee the code review process. These capabilities help ensure Everi Games always produces robust software that complies with the strict regulations of the gaming industry.

## Whirlpool — Improving dev collaboration across continents

Whirlpool is the number one major appliance manufacturer in the world. The company markets appliances in more than 170 countries under the Whirlpool brand as well as KitchenAid, Consul, Maytag, Brastemp, Amana, Bauknecht, Jenn-Air and Indesit.

Ajay Tamboli, an Electronics Tools and Infrastructure Lead for Whirlpool, develops software as part of a team of 150 electronic software engineers. He explains the importance of software quality for the company.

*"When developing and reviewing software code, it's critical that we identify any bugs early in the process, well before the code gets embedded into the control panels of the appliances. Once an appliance is shipped to a retail store or customer, any changes to the code require an on-site visit to change out the control board, which impacts customer satisfaction."*

Whirlpool's commitment to quality is dependent on their ability to have team members, located across continents and time zones, collaborate effectively.

SmartBear Collaborator helps the Whirlpool software development team work together to produce high-quality code. The solution allows the team to peer review code as well as user stories and test plans in a transparent, collaborative framework — while also keeping the team up to speed in real time on code changes. By enabling team members to work together to review their work, Collaborator helps Whirlpool catch bugs before embedded software is installed on appliance control panels.

*"With Collaborator, the entire team can smoothly interact*

*on multiple projects, even if one team member is working from India with others working from North America and Europe. We no longer have to coordinate manual reviews involving spreadsheets and conference call at odd hours — each of us can communicate code review changes on our own time."*

Key Takeaway: To coordinate code reviews among 150 electronic software engineers spread across the globe — so they can efficiently identify software bugs before code is embedded into appliance control panels and shipped to customers — Whirlpool turned to SmartBear's peer code review tool, Collaborator. The solution helps the software development team work together to produce high-quality code. This accelerates the overall development process and delivers software to market faster while also resulting in better-performing appliances for Whirlpool customers.

## Cisco Systems — Changing internal perceptions on peer code review

After implementing an engineering-wide policy that mandated code review for every bug fix before check-in, the company began looking for a code review tool. Cisco wanted to increase defect detection, to simplify and speed up the review process, and remove some of the drudgery normally associated with code inspections. Brandon Fuller, Manger, Software Development, Cisco Systems, explains:

*"Our team is geographically distributed across the US from coast to coast, so a solid communication infrastructure is essential for us to work together. Multiple other teams at Cisco (about 50 people total) use Collaborator to help groups communicate from Belgrade to San Jose to Research Triangle Park and Boulder. With team members in so many different time zones, getting together for live reviews rarely happens, so we needed a code review tool to make it easy for developers to carry on discussions with each other, but still review code in their own time."*

The team at Cisco needed a tool that allowed for cross-location collaboration, but also needed something that would fit into their existing systems. By integrating the tool with their existing defect tracking system, they were able to improve collaboration without having to manage their workload in complete separate environments.

*"This system will not let us resolve a bug until the code has been reviewed, and we must submit reviews to this system in a certain format. We were able to work with SmartBear to integrate Collaborator into our defect system to make this process smooth. Now, when a team member finishes a review, the right "enclosure" for our system gets built automatically and attached to the defect."*

Prior to implementing the tool, code reviews were only done on a project basis. By choosing a tool that allowed team members to collaborate without being in a single location, Cisco has been able to make code reviews a standard practice on their development team.

# Bringing it all together

**While there is no singular definition of "code quality," there are steps teams can take to improve quality and reduce costly defects in the development process.** Code review will be a critical component of your code quality strategy, but you will also need to consider your coding standards, requirements, of the code that's being written.

**Collaboration is crucial at each phase of the development process.** Collaboration takes place with idea creation, user stories, product requirements, source code, test artifacts and ultimately deployment.

**One of the most important components of a collaborative approach to software development will be code reviews.** An effective code review process requires a tool that enables collaboration across teams and across locations.

A dev collaboration tool like Collaborator has helped industry-leading organizations like Cisco, Everi Games, and Whirlpool improve code quality and can integrate perfectly within your development team, regardless of systems or processes that you're currently using.

## Dev Collaboration for Your Entire Organization

FREE 30-day trial of Collaborator

# Collaborator is the industry leading peer code and document review tool that:

- Encourages teams to communicate daily
- Helps bring new team members up to speed
- Aids in finding defects earlier - before they ship to customers
- Provides metrics and reporting for regulatory compliance and internal/external audits



# SMARTBEAR

Over 4 million software professionals and 25,000 organizations across 194 countries use SmartBear tool

**4M+** users

**25K+** organizations

**194** countries

See Some Succesful Customers >>

## API READINESS

Functional testing through performance monitoring

SEE API READINESS PRODUCTS

## TESTING

Functional testing, performance testing and test management

SEE TESTING PRODUCTS

## PERFORMANCE MONITORING

Synthetic monitoring for API, web, mobile, SaaS, and Infrastructure

SEE MONITORING PRODUCTS

## CODE COLLABORATION

Peer code and documentation review

SEE COLLABORATION PRODUCTS


Collaborator