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

SMARTBEAR

Contents

Introduction

Every year, we hold a State of Code Quality survey to discover the best way to improve software quality. Every year, we rediscover the key to improvement: reviewing.

Whether it's code review or document review, the best way to ensure quality is to make sure your development, testing, and business teams are all on the same page.

Code review is a critical component to code quality, but in an increasingly agile world, collaboration on development teams needs to go beyond peer code review. Even with code reviews, you'll find bugs if you don't have clear standards for the code you're writing, and an understanding of the user requirements.

Software quality is driven by collaboration throughout the development lifecycle. Tools that support this process include integrated development environments, build and test automation tools, requirements management and bug tracking tools, and peer review tools.

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

This ebook will discuss the importance of code quality and how it fits into dev collaboration.

- **Defining code quality**
- **Understanding the role of collaboration**
- **Improving code quality through a collaborative approach to reviews**
- **Overcoming obstacles to peer reviews**

Defining Code Quality

No matter what programming methodology your team adopts – waterfall, scrum, test driven, extreme, or Agile – maintaining a standard for your code base is vital for project momentum. It will also address personal biases when attempting to define ***"good code."***

Establishing coding standards

No matter what programming methodology your team adopts, maintaining a standard for your code base is vital to project momentum. This will also help address personal biases.

Establishing code standards, either through a doc or by adding a coding formatter style sheet to a preferred IDE, improves version control and takes ego out of the mix. Removing coding styles makes the end result (your code base) seamless, and alleviates concerns that one developer is changing another developer's code.

Using style sheets for comparison provides clarity and ownership as a group to the entire team. It also reduces the chance that you'll need to clean up after a code import causes issues in readability and functionality.

Meeting the Requirements

Another measure of quality is whether the code meets the customer's needs. Under this definition, if code is beautifully written but fails to accomplish the intended outcomes, it can be described as low quality.

Of course, even software that meets all of the customer's requirements could still be poor quality. There may also be specific needs related to marketing or performance, as well as regulatory requirements within certain industries. Software typically changes as new customer requirements and desires are added. Therefore, code must be easy to change and improve.

Maintainability

When you build software, the new code interacts with the old stuff: calling into it, relying on it, and running beside it in production.

Old code has an impact on your team. If your code base is difficult to maintain and update, it's high in technical debt. It therefore has a higher total cost of ownership, because changes take longer to implement, and there's a greater risk of introducing defects while adding new features.

Freedom from Deficiencies

There are many aspects of code that can be observed and assessed. Freedom from deficiencies refers to well-performing non-functional requirements such as maintainability, readability, and scalability. With so many qualities of code to consider, it's necessary to decide which aspects to optimize, and which to monitor less aggressively.

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

Reducing Time-to-Context

Time-to-context is the time a developer must spend to get enough context to make changes. One advantage of maintainable code is that it reduces time-to-context for the code reader. This can be achieved with simple designs and a clean separation of concerns, as well as ensuring each class has a single responsibility. Maintaining low technical debt and a fast time-to-context for the reader ensure that code can be easily maintained. The simpler code is, the fewer defects it will have.

The Reality of Code Quality

Over time, respondents to our annual State of Code Quality surveys have felt positively about the quality of the software they help build. In aggregate, only around 10% felt negative about the software they help develop. The overall positive sentiment was felt across industries and roles within software teams.

But while two-thirds of respondents are satisfied with the quality of software they help build, there's still room for improvement. This is especially true on larger development teams. One in five respondents on development teams with 50+ employees disagreed or strongly disagreed that they were satisfied with the quality of software they help build.

Achieving and maintaining a high standard of quality depends on collaboration. But what does collaboration mean in a modern development team?

Collaboration in Pursuit of Quality

Collaboration is critical at every stage of the development process and post-COVID, this factor became even more clear. Improving collaboration leads to added speed, quality, and agility. Developer collaboration encompasses many areas: idea creation, user stories, product requirements, source code, test artifacts, and ultimately deployment.

The idea generation phase may need little more than a whiteboard. As user stories and requirements develop, though, it's important to track changes and provide feedback. For some regulated industries, development procedures require audit trails. Documenting the conversations around requirements is as critical as later stages of product development. When future teams need to review and make changes to existing legacy code, they can refer to these documents and understand why decisions and changes were made.

All companies now write software. Collaboration should be a top priority.

In many cases, collaboration starts with establishing a documented code review process. This ensures teams and organizations are releasing high quality software downstream to test engineers and, ultimately, customers. The tools needed for code review vary, but it's driven by a need to release code quickly with as few defects as possible.

Most organizations also pursue distributed development – a reality made more true following COVID-related quarantines and remote job roles. These distributed global teams require a way to easily provide feedback on source code with shortened feedback loops due to time zone challenges.

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

Collaboration doesn't end after the code is written.

Both 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 might include:

- **Test plans**
- **Test cases**
- **User scenarios**
- **Requirements**

When the entire team knows what a successful test process looks like, quality can be maintained. Many deployment teams forget to review their artifacts, but reviewing infrastructure code is critical to making sure 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 argue that their workload is too heavy to let them participate. Yet it saves time and money, and lightens the workload, because defects are easier to find earlier in the process. As you get further from user stories/requirements and move 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:**

- **Bring new team members up to speed faster**
- **Find defects earlier** – before they ship to customers – which means it's cheaper to fix them early
- **Create audit trails**
- **Minimize risk due to turnover**
- **Increase knowledge share** – more than one individual knows what is going on
- **Ensure distributed teams communicate daily**
- **Facilitate faster onboarding**

Why code review is an essential part of software collaboration.

This collaborative approach to software development often begins with peer code reviews. When a team does code reviews, everybody understands the codebase and its data flow. More importantly, everyone understands the reasoning behind the decisions for changing the code, not just the mechanics of the code. For instance, if everyone understands this application needs to be lightning fast, the code will be created and reviewed with that priority in mind. This makes it easier to make choices across the project, since everybody is on the same page.

A person uninvolved in the project should be able to read a portion of code and understand what it does, and see the constraints and preconditions of its use and contextual information. (For example, 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 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 report also cited the importance of sharing knowledge across the team, the ability to mentor less experienced developers, and increased collaboration as benefits of code review.

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

How Code Reviews Help Improve Code Quality

As a developer, your challenge is in ensuring that expressions of intent are easily understood by others, while meeting quality standards.

The reasons for code reviews are as diverse as the environments in which they are conducted. However, most code reviews have these goals:

- **Defect-free, well-documented software**
- **Software that complies with enterprise coding standards**
- **Knowledge sharing between developers**

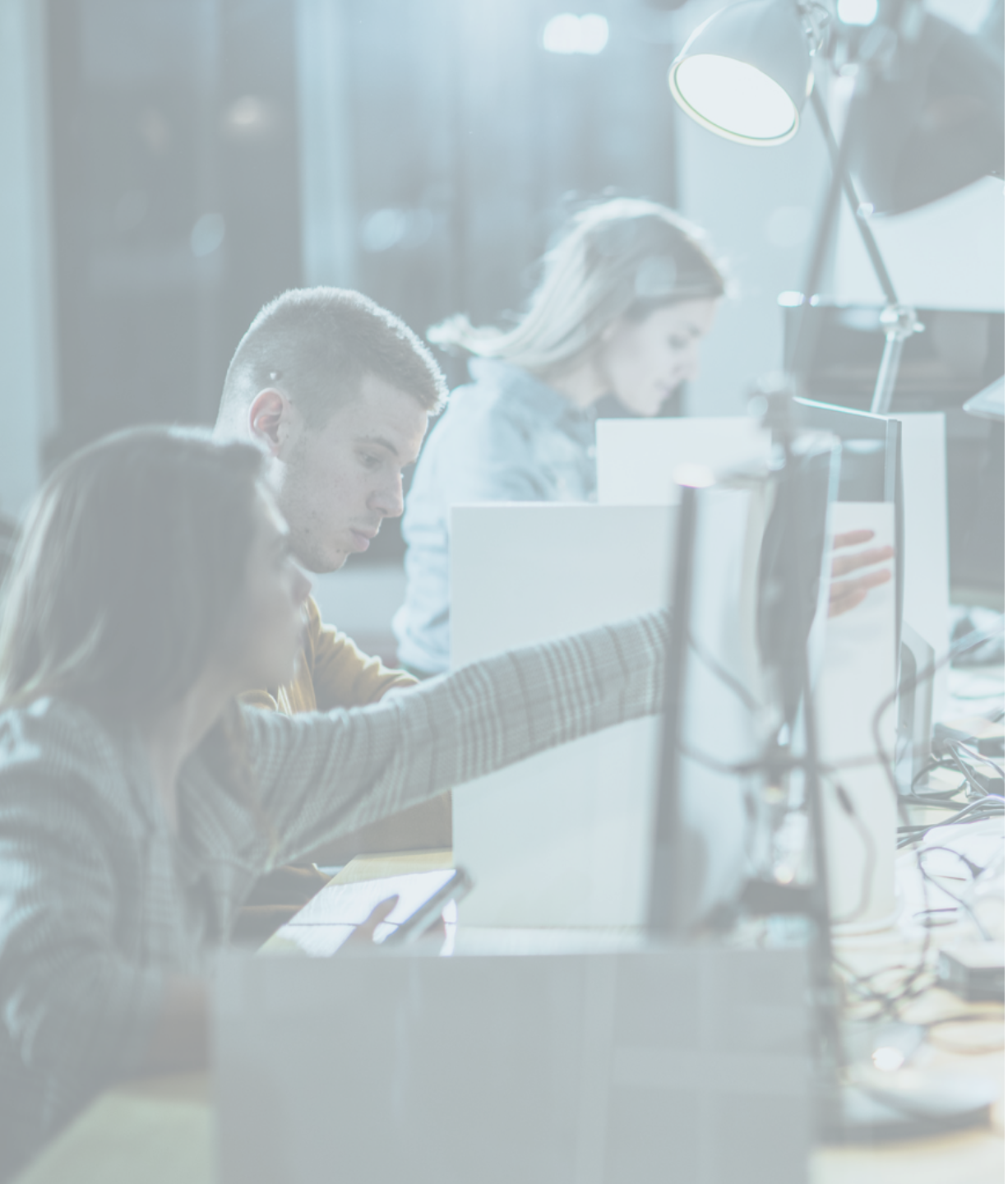
Other objectives include maintainability, security, consistent end-user documentation, adequate comments in code, complete unit tests, and scalability.

How code review works

Code review involves one or more *developers examining source code they didn't write* and providing feedback, both negative and positive. This ensures the code is readable as well as maintainable even by people unfamiliar with that project. Typically, reviewers have a standard checklist for effectively performing code reviews, so they can find defects and 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, which are designed to be written quickly and most likely not used in production. Even during the final stages of development, code review reduces the number of regression bugs and ensures that company coding practices are followed.



Improving code quality with code review

Just because your code works, doesn't mean it's actually right.

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 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 2020 State of Software Quality | Code Review report understood that code review was the most important way to improve code quality, **they also acknowledged challenges.**

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

It's common for teams to adopt an ad hoc approach to code review, in order to manage their workload. While this can be useful on certain projects, it has the potential to become a blocker in your development process.

In fact, when we asked participants how frequently they were doing ad-hoc code review, fewer than 5% said they did code review daily. On the contrary, 23% of teams that were using a tool for code review were able to do it daily.

Here's an overview of the key tools and unctionality of tool-assisted code reviews:

1. **A central location to review and discuss user stories, requirements, code, and test plans.**
2. **Comment and defect creation, and the ability to verify changes have been made.**
3. **Customizable workflows configured for each team in the enterprise.**
4. **Ability to review documents, images, and source code in their own unique workflows.**
5. **Reporting and audit trails for regulatory and 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 abilities for easy reference to issues.**
9. **Configurable roles and responsibilities for the review to ensure the right people are contributing.**
10. **Ability to run scripts based on events in the review to trigger actions in third party utilities.**
11. **Integrations of different SCMs make it easy to attach files to reviews.**

Collaboration Is Action

Users of the SmartBear tool Collaborator are more able work together to improve code quality and address business challenges. The benefits for these organizations go beyond the quality of software they deliver.

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

Everi'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 studios in Chicago and Reno.

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

> "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 fulfil 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 racetracks, 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 now work together more efficiently, and their managers more effectively oversee the code-review process. These capabilities ensure Everi Games 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, 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 depends on their ability to have team members collaborate effectively even when distributed across continents and time zones.

Collaborator helps the Whirlpool software development team produce high-quality code. The team peer reviews code, user stories, and test plans in a transparent, collaborative framework, while also keeping 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 around the globe, so they can efficiently identify software bugs before code is embedded into appliance control panels and shipped to customers, Whirlpool turned to Collaborator as well. The tool 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, Cisco Systems searched for a code-review tool. They wanted to increase defect detection, simplify and speed up the review process, and remove some of the drudgery normally associated with code inspections.

Brandon Fuller, Manger of Software Development at Cisco Systems, explains:

"Our team is geographically distributed across the US, 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 and would fit into their existing systems. By integrating the tool with their defect tracking system, they improved collaboration without having to manage their workload in 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."

Before 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 made code reviews a standard practice on their development team.

Bringing It All Together

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

Collaboration is crucial at each phase of the development process – especially in a time when common workplaces are hard to come by. 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.

With the right tools and best practices,
your team can peer review all of your code.

Start Your Free Trial Today


SMARTBEAR