Code Review in Distributed Teams: The Engineering Manager's Guide to Peer Code Review Culture

How serious are you about the quality of your work? If the mere mention of the word ‘feedback’ gives you chills, it might mean your team isn’t fully benefiting from your experience and perspective as a manager.

In software development, feedback shouldn’t be a dirty word, but a practice that signals maturity and a strong commitment towards becoming a better software engineer. And according to Karl E. Wiegers (Engineering Manager & Consultant, who writes about process improvements), feedback mechanisms are a clear indicator of a healthy software engineering culture that creates the premises for higher productivity and better product quality.

One obvious feedback mechanism is code review, the process by which someone other than the code’s submitter checks a program by looking at parts of its source code. It’s both a technical activity, a social interaction, and the single biggest thing you can do to improve your team’s code.

Nowadays, building a software product involves teams that collaborate across corporations, time zones, continents, cultures, and

languages. It's become increasingly common over the last decades and the past year, especially in the context of the COVID-19 pandemic.

And these geographically dispersed teams aren't making code reviews any easier. In fact, geographic distance can impose additional challenges to the reviewing process. One study on the effectiveness of peer code review in distributed software development found that *"the number of involved teams, locations, and participant reviewers generally improve reviewer contributions, but with a severe penalty to the duration."*

The good news is that tool support and asynchronous communication can help deal with geographic distribution and large teams. It's something that we, at Waydev, have focused on, in our path to help software development teams (whether they're distributed or not) and their managers effectively review their team's work and speed up the development process and deliverability.

We believe in a peer code review culture and the benefits it can yield in Agile software development. That's why this article will highlight

some of the best practices in a code review that you can implement to enhance and update your review workflow.

By the end of this article, the insights you'll have gained might help your peer review program succeed where others have failed. Let's get into it.

Why peer code review is important for software development teams (distributed or not)

More and more engineering teams have started to adopt peer code reviews as part of their development process—and for good reason.

Peer code review adds a much-needed collaborative element to the development phase of the software development process. If reviews can improve the quality of papers, novels, and design, then it only

makes sense that they would and should improve the quality of code.

Peer code reviews are one of the most powerful software tools available. They've been proven to reduce cycle time, improve software and code quality, as well as save developers and team leaders time in the long run.

Check out this quote by Karl Wiegers, software engineer and author of "Humanizing Peer Reviews":

"Peer review methods include inspections, walkthroughs, peer desk checks, and other similar activities. After experiencing the benefits of peer reviews for nearly fifteen years, I would never work in a team that did not perform them."

Code reviews can benefit every type of team, regardless of size or development methodology. But distributed engineering teams can get even more from this process since their work is decentralized across the team. In an Agile distributed team, peer code reviews help facilitate knowledge sharing across the code base and across the team.

Having team leaders engage their colleagues across the globe to improve the quality of their work and increase their performance is a sign of a healthy software engineering culture. And the very best software engineers understand that reviewer input is part of what makes them excellent at what they do.

Peer code review benefits

For team leaders and managers who want to deliver quality products, and feel pressure to do so quickly, peer code reviews can do wonders in terms of shortening the product development cycle time, reducing field service, lifetime maintenance, and customer support costs, and gaining earlier insights into project risks and quality issues.

This allows for more resources for new development projects and a general improvement in teamwork, collaboration, and development effectiveness.

For developers, this can mean less time spent performing rework, learning better techniques from other developers, experiencing an

increase in programming productivity and a decrease in unit testing and debugging time, during integration and system testing.

Engaging in peer code review can enhance performance and yield specific benefits such as:

- Dead code elimination – Great managers and team leaders understand that time spent reviewing code isn't time wasted, especially when other team members willingly reciprocate and work together towards a mutual goal: better code.
- Improved code quality – In a peer review, someone from the development team examines the piece of code for quality problems and improvement opportunities. Recognizing that team success depends on helping each other do the best job possible means having peers—not customers—find defects.
- Optimized engineer collaboration – Code reviews can motivate developers to do better and practice superior code craftsmanship, since they know their colleagues will closely examine their work.
- Consistency in a code base – This makes code easier to read and understand, helps prevent bugs, and facilitates collaboration between team members.

Many team leaders have already seen improvements to their overall Agile development experience. The productivity of Scrum teams has also improved after they've implemented code review as a

requirement while reaping the resulting benefit: higher-quality software.

Here's what these benefits mean in terms of three of the most important aspects a manager deals with on a daily basis: time, money, and their team.

- Peer code reviews save time by:
    - streamlining the development process upfront
    - drastically reducing the amount of work required of QA teams later .
- Peer code reviews save money by:
    - catching the types of bugs that might slip undetected through testing, production, and into end-users' laptops.
    - sparing customers from developing a negative perception of your product and your capabilities as a development team.
- Peer code reviews build company culture by:
    - promoting openness and encouraging programmers to discuss their code
    - improving team collaboration by creating collective ownership of the source code, which results from collaborative, rather than individual work
    - making sure all of your engineers share a similarly high set of best practices
    - fostering knowledge sharing that benefits both submitters and reviewers
    - enhancing security by requiring engineers to constantly update their security knowledge and to practice it inside the implementation of the product itself

- enhancing junior developers' educational process, by having senior team members demonstrate better ways to write clean code, solve common problems, and visually identify any number of potential troublespots, such as memory leaks, buffer overflows, or scalability issues.

How to optimize your code review process

Effective code reviews can be invaluable to the Scrum methodology. These code review best practices will help team leaders and managers effectively review work and increase velocity and deliverability, in the context of a distributed software development team.

We've split the practices across the stages of a code review workflow: before, during, and after the code review. Our goal here is to define tasks that are granular enough so that the code requiring review is in manageable chunks and does not overwhelm the review process.

Before the code review:

Decide and standardize the right moment for reviews.

Taking the time and effort to put together a code review strategy—and consistently following it—is worth it, in the long run. This starts with standardizing when and where code reviews should take place.

While there are many places to go to for code reviews (from colleagues or team members or companies that provide software audit services to freelancers, agencies, and even programming subreddits for smaller code fragments), you can decide at what stage of the project to ask for code reviews—we recommend doing it before merging the code to the mainline branch.

Requiring code review before merging upstream ensures that no unreviewed code slips through. Which means that bugs and bad design patterns are caught before they enter your codebase and

have a chance to make a lasting and regrettable impact on your product.

Have your team prepare in advance for reviews.

This applies to both submitters and reviewers, especially if your code review workflow involves formal review meetings. For team members that are looking for code reviews, make sure they complete, test, and review the code themselves before submitting.

Reviewers should examine the material ahead of time and identify the issues they want to raise. Without this preparation, you risk people spending the meeting time doing all of their thinking on the spot and likely missing out on important issues.

During the code review:

Include everyone in the code review process

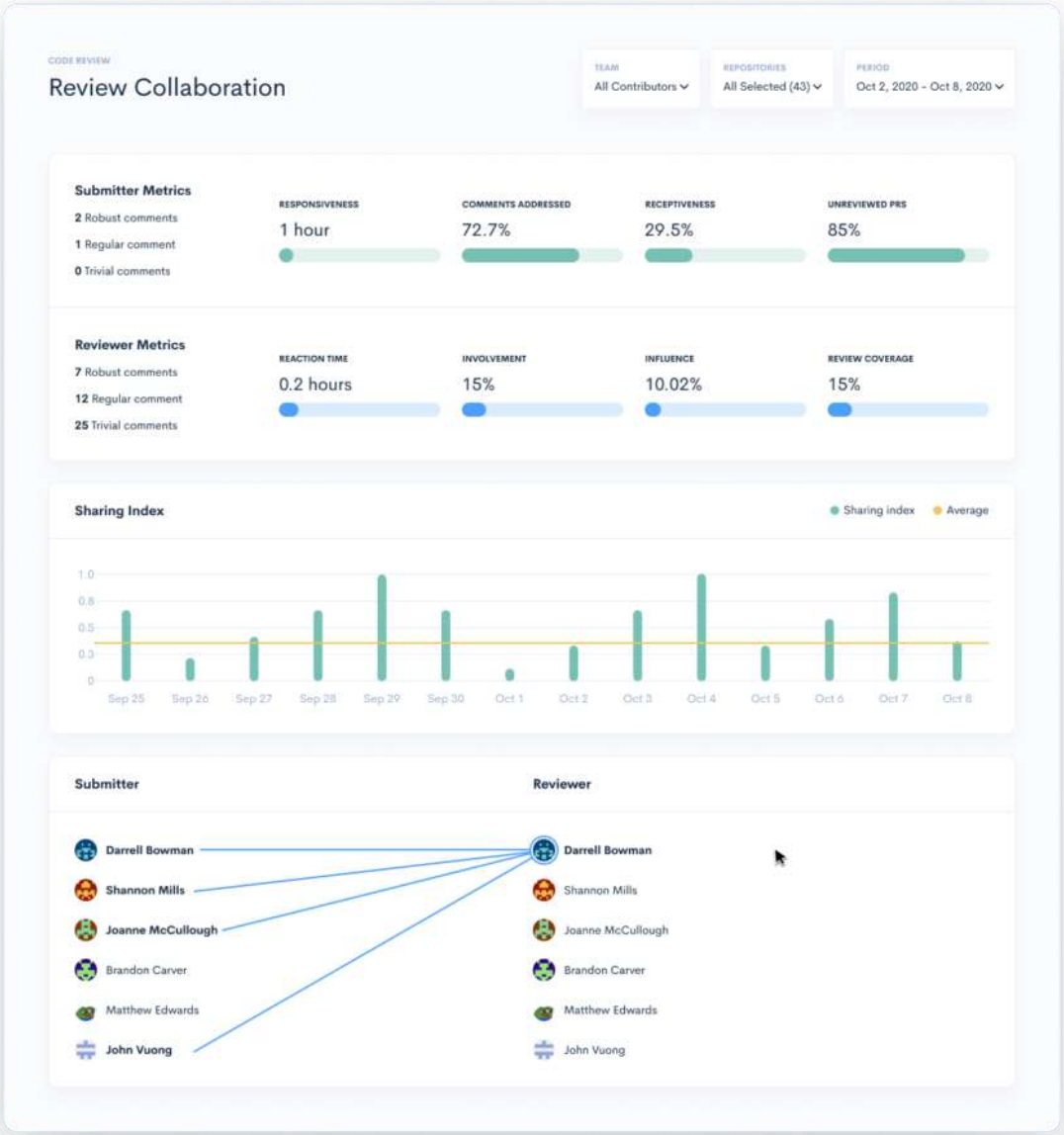
Engineers can play the role of both submitters and reviewers, especially in a Distributed Engineering team. No matter how senior

the programmer is, everyone should have a chance at reviewing and being reviewed. Everyone performs better when they know someone else will be reviewing their work.

However, asking someone to tell you what's wrong with your work isn't instinctive behavior. It takes time for a software organization to instill peer reviews into its culture and you need to make sure everyone's doing their part.

To streamline this process, you can try Waydev's Review Collaboration feature to see code collaboration stats between the submitters and the reviewers and understand how your engineering teams work collaboratively.

Use the Sharing index report to see a visualization of the ratio of active reviewers to submitters. Then, use the Collaboration Map to check on your senior engineers and see whether they are active in the code review process.



Take cultural differences into account.

In Distributed Engineering organizations, collaborations often take place across great distances and involve people who work in

different cultures. These cultures can have a different view of making critical observations about someone else's work. Some team members might not feel comfortable critiquing at all. This is a good way to avoid hurting someone's feelings but not a good way to improve a software product.

When planning reviews for cross-cultural development projects, be aware of these interaction differences and how they will affect the review process, and try to create safe environments for people to raise potential defects without making anyone uncomfortable. Consider which approach will work best when the team members are separated.

Make sure reviewers are using the right language.

If there were one rule to follow during peer code reviews, it would be this one: critique the code, not the submitter. When code review results are used to measure the caliber of the developer, rather than the quality of the code, this can lead to team level tensions.

As a team leader, you can avoid these by making sure both submitters and reviewers are using the correct language and tone, which can have a profound influence on how developers accept and adopt the code review guidelines. This might mean avoiding possessive pronouns, posing questions in a less threatening way, and asking for clarification rather than assuming ignorance.

Decide and standardize how to split code fragments.

This goes back to the standardization aspect. Changes to the source code can differ in scope and size (your team might be looking to get rid of a bug or add/remove a feature) and it's up to you to decide how to split reviewable fragments—shorter ones are always easier to work with and yield better results.

Have reviewers be mindful with refactoring.

When performing code reviews, remember that the focus is on improving internal structure without altering external behavior. It's

common sense to clean up after oneself and even more so when coding—keep it clean and elegantly simple.

Put some effort into your commit message.

A commit message is a note that signals changes are made to a code repository. Make sure it's concise and easy to understand by anyone and everyone involved in the process—keep it clear and short; use bullet points, asterixis, hyphens, and hanging indents.

Automate what can be automated.

The best use of a developer's time is reviewing qualitative aspects of code, such as logic, design patterns, and software architecture. When it comes to quantitative, human review time can become expensive, and automation tools are the obvious better choice.

You can use linting tools to take care of style and formatting conventions, Continuous Quality tools to catch potential bugs, anti-patterns, and security issues, and a Git Analytics platform like Waydev that uses metrics to optimize engineer collaboration across

code reviews. Our new Agile data-driven method of tracking engineering teams' output directly for your Git repos, without any manual input—which is the whole purpose of automation.

Most of these code review tools integrate well with code hosting platforms and can enhance your overall code review workflow.

After the code review:

Optimize your review workflow

Do what you do best but even better. Identify long-running pull requests, unreviewed pull requests that have been merged, and spot-closed pull requests that have not been merged. Solving these issues will go a long way towards more effective peer code reviews.

Look at the right code review metrics.

Here are some of the main code quality metrics that you should check when performing a code review:

- Code readability – the ease with which the software is read and understood.

- Maintainability – how easy it is to incorporate the alterations later on, while taking into account the prospect of malfunctioning the entire application.
- Extensibility – how changeable and extendable the code is, in terms of incorporating advanced features into newer versions without disturbing the overall program and software functions.
- Clarity – how understandable and comprehensible the code is, as engineers read as well as work on it in the various phases of development.
- Efficiency – the number of assets that are consumed to build a code, as well as time taken to run the code.
- Documentation – the degree to which the submitter explains every single method and component used, along with the logic behind the various programming alternatives.
- Low technical debt – the cost of refactoring a piece of code or system to keep it working efficiently.

At Waydev, our code review workflow reports paint a fuller picture by taking into account some additional metrics, which deal both with the reviewer, as well as the submitter:

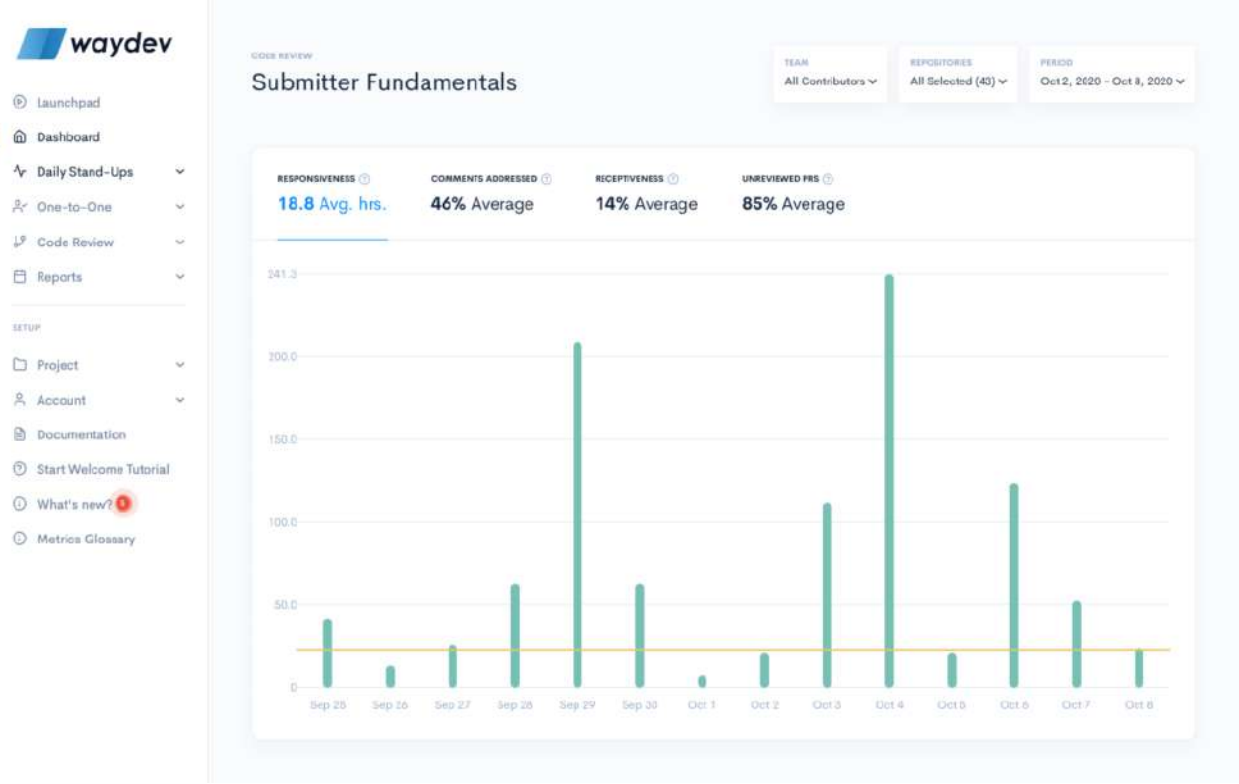
Submitter metrics:

- Responsiveness
- Comments addressed
- Receptiveness
- Unreviewed PRs

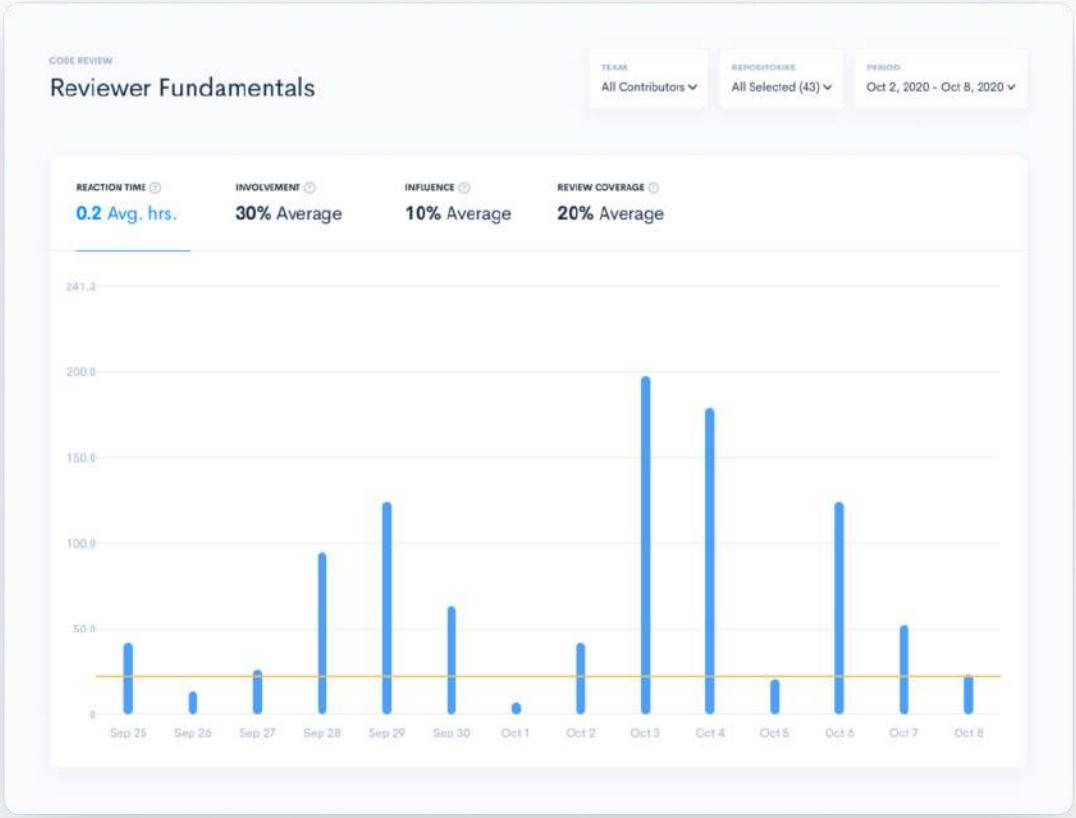
Reviewer metrics:

- Reaction time
- Involvement
- Influence
- Review coverage

Submitter:



Reviewer:



Incorporate these metrics into retrospectives.

Now that you have an overview of how your team can handle peer code reviews, it's time to take some action. Your position as team leader means you're able to facilitate conversations on the gap

between your team's objectives and actual results, then determine which actions, if any, to take.

Start incorporating code review metrics into your retrospectives. It will help identify when processes need to be adjusted and make coaching opportunities easier to identify. That way, you and your team can work together towards a more collaborative, productive, and engaging code review process.

Wield automation to enhance your code review workflow using Waydev

Use the suggestions in this article to help your team maximize the return on investment they make in code reviews. The teams we know that have adopted code review protocols agree that every minute they spent on it was worthwhile.

Then, if you're serious about maximizing the quality of your software, make sure you're using every tool in your arsenal. Waydev is a great way of learning what is going on in the code review process and using data to optimize engineer collaboration across distributed teams.

These features help managers get that data and make it work in their code review process:

- Fundamentals – Determine if your code review workflow objectives are on track.
- Review workflow – View a map of pull request activity in the selected time frame. Identify long-running pull requests, unreviewed pull requests that have been merged, and spot closed pull requests that have not been merged.
- Review collaboration – Understand how your engineering teams work collaboratively. Effectively communicate the healthy tension between speed and thoroughness in code review.
- PR Resolution – Identify the bottlenecks in your PR cycles over the course of the sprint. Find outliers, visualize high-level team dynamics and the underlying activities that can contribute to those dynamics.

Want to know what Waydev can do for your Code Review Workflow?

Schedule a live demo and see for yourself.

About Us

Our mission is to provide engineering leaders with a way of measuring the performance of their engineering teams. We strive to help the technology industry move towards a data-driven agile development methodology and make decisions supported by data.

We are trusted by **Fortune 500 companies**, such as Blue Cross Blue Shield, TATA, and Carrier, and we are also loved by startups (#1 Product on Product Hunt).

Waydev is the **G2 Market Leader** in Winter, Spring and Summer 2022

Visit waydev.co to learn more

waydev

TRUSTED BY **FORTUNE 500** COMPANIES