Team collaboration metrics

Tags: Flow

The Team collaboration metrics are designed to give managers a bird's-eye view of how their teams are behaving in the code review process. These metrics illuminate knowledge concentration centers, code that may require your attention, and bottlenecks to a pending release.
These are some of the metrics that comprise the Team collaboration metrics:

- **Time to merge:** The average time it takes to merge a pull request.
- **Time to first comment:** The time between when a pull request is opened and the time the first reviewer comments.
- **Number of follow-on commits:** The number of code revisions once a pull request is opened for review.
- **Raw activity:** A raw count of comments and follow-on commits associated with a PR. As a pull request ages, its raw activity does not change and will always stay the same. Pull requests are ordered by the raw count of comments and follow-on commits in an increasing or decreasing order.
- **PR activity level:** A measure of how active a pull request is on a scale of Low, Modest, Normal, Elevated, High, as calculated by the comment count, word length, and recency of comments.
- **Recent ticket activity level:** A measure of how active a ticket is on a scale of Low, Modest, Normal, Elevated, High, as calculated by the comment count, word length, and recency of the comment.
- **Knowledge Sharing Index:** Measures how broadly information is being shared amongst a team by looking at who's reviewing whose PRs.
- **Number of PRs reviewed:** Total number of PRs that were reviewed.
- **Number of authors reviewed:** Total number of submitter users that were reviewed.
- **Available reviewers**: The count of unique users who committed or commented in the selected time period.
- **Active reviewers**: The count of active users who actually reviewed a PR in the selected time period.
- **Submitters**: Total number of users who submitted a PR in the selected time period.

These metrics are designed to promote healthy collaboration and provide prescriptive guidance to improve the productivity of the team’s code review process as a whole.

As with any data point, these metrics should be used in context. “What’s right” and “what’s normal” will vary depending on your team’s situation.

Time to merge

Time to merge is the average time it takes to merge a pull request.

As a manager, you just want to drive this number down. Pull requests are a period of review. They should open, there should be discussion, perhaps some follow-on commits, and they should be merged.

Long-running PRs are something you should manage relatively aggressively. If you notice long-running PRs, use the **PR Resolution** report to identify the root cause and take action accordingly.

Learn more about Time to merge.

Time to first comment

Time to first comment is the average time between when a pull request is opened and the time the first reviewer comments.

This is another metric that, as a manager, you want to drive down. When someone opens a PR, you want to see people getting feedback in early. It speeds up the entire process that follows: responding to comments, incorporating feedback, getting those changes reviewed, and so on.

Keeping this number low is a team effort. You can encourage the team by ensuring them that it doesn't matter who gets assigned to the PR, it matters that the team acknowledges the open PR and gets on it as soon as possible.

Learn more about Time to first comment.

Number of follow-on commits

Number of follow-on commits is the average number of code revisions once a pull request has been opened for review.

This helps gauge the strength of your code review process. If nothing ever changes because of the review process, then why are you

doing it? If every PR yields a lot of follow-on commits, then perhaps more planning and testing is in order.

Generally speaking, this is a Goldilocks number (eternal site, opens in new tab). This means there's a happy middle ground based on your team's culture. You want to manage this number as it changes, rather simply up or down. Large or sustained changes in this average suggest a change in the team's dynamic that should be investigated.

Recent PR activity

Recent PR activity is a measure of how active a pull request is on a scale of: None, Low, Modest, Normal, Elevated, and High. The measure considers the volume of commenting and follow-on commits in a pull request.

Recent PR activity has a built-in decay function, so older comments are scored lower than recent comments.

This metric gives managers a way to gauge how much chatter is happening around a PR without having to read the actual comments of the PR. This is particularly useful coupled with Time to merge. A long-running PR coupled with Elevated activity suggests disagreement or uncertainty and often warrants a manager's attention.

Generally speaking, you're looking for outliers. When you find long-running, low activity PRs you want to nudge folks to approve or close the PR. PRs that have a lot of activity should be reviewed carefully for both tone and content. This helps you understand if discussions are degenerating into arguments or whether they're driving toward resolution instead of going in circles.

Recent ticket activity

Recent ticket activity is a measure of how active a ticket is, as calculated by the comment count, word length, and recency of the comment.

Recent ticket activity has a built-in decay function, so older comments are scored lower than recent comments.

As concepts, ticket activity and PR activity are very similar: both look at where people are spending time in the review process.

This metric gives managers a way to gauge how much chatter is happening around a ticket without having to read the actual comments of the ticket. This is particularly useful coupled with Cycle time. A long-running ticket that also shows Elevated activity suggests disagreement or uncertainty and often warrants a manager's attention.

Generally speaking, you're looking for outliers. When you find a ticket with a lot of activity late after an engineer begins implementation, you should review it carefully for potential problems that could impact timely delivery.

Notably, you want to watch for tone, content, and most importantly, material scope creep.

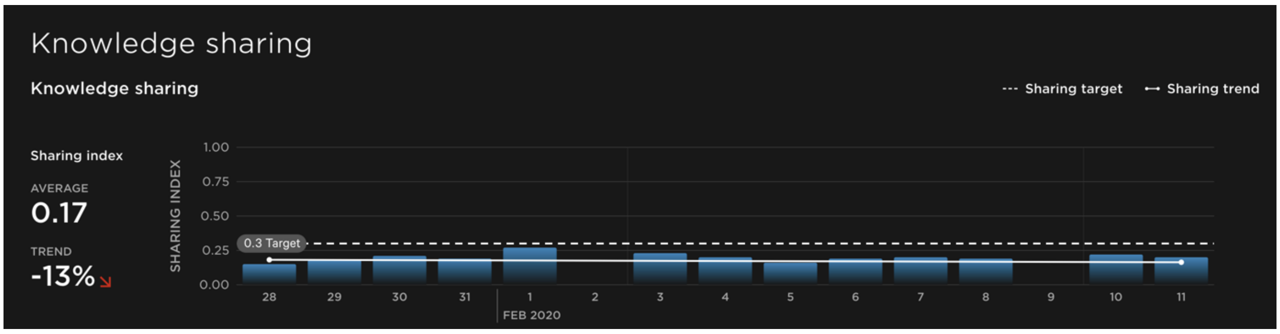
Tickets usually represent the source of truth for what is to be built. You should expect lots of comments prior to implementation when a feature is being designed or a requirement is being documented. However, after implementation begins, you should expect very few comments with only minor clarifications in requirements, normal status changes, etc. When you see Ticket activity creep up, it's often a sign that a manager can add value by intervening to make sure the ticket stays on track.

Knowledge sharing index

Knowledge sharing happens when engineers are familiar with multiple areas of the code base. There are two main ways to increase your team's knowledge sharing. Engineers can work on multiple areas of the code base, or engineers can review each other's code.

Since it's not always practical for an engineer to work on multiple areas of the code base, reviewing code is a valuable opportunity to engage with different areas of the code base. This allows engineers to know more parts of the code and ensures no one person is a knowledge silo for a particular part of the code.

The **Knowledge sharing index** measures how thoroughly team members are sharing information via code review. Use the Knowledge sharing index to view which engineers are reviewing others' pull requests as well as who owns those pull requests. This helps you understand if you are increasing the number of people participating in code review or if only a few individuals are reviewing code.



The Knowledge sharing index measures knowledge sharing on a scale of 0-1.

- 0 represents the least possible knowledge shared, meaning one person is doing all the reviews.
- 1 represents an even distribution of the work.

We suggest aiming to be at or above 0.6. If you are below 0.3, you should make knowledge sharing a focus for the team.

Flow uses a variation of the Gini Coefficient (external site, opens in new tab) to calculate the sharing index. The Gini Coefficient comes from the field of economics and is commonly used to measure the wealth distribution in a society. When you hear TV pundits say things like “the top 1% owns 40% of the wealth of a country” they are resting on this field of research.

In our case, think of a PR comment as a unit of wealth. In a perfect world, everyone performs an equal number of reviews and the wealth is evenly distributed. Alternatively, if one person performs all the reviews, they have hoarded all the knowledge available via reviews.

In practice, PR reviews are not the only way to share knowledge, just as there are other forms of wealth that aren’t considered in a Gini Coefficient. However, we have found that the Knowledge sharing index makes for a good signal for management. You can use it to help identify engineers who are not part of the everyday flow of reviews, as well as find silos and other anti-patterns that can bubble up because they're hard to spot. The Knowledge sharing index adds a nice tool to help your team collaborate and share knowledge via reviews.

Look for outliers and stranded engineers, then take action to help get them involved. Use the Knowledge sharing index to manage your team’s broader trend toward or away from that imagined ideal of shared knowledge, keeping in mind that perfect distribution is rarely achievable nor is it desired.
