Getting Faster Pull Requests in an Agile Environment

By Steve Berczuk - May 14, 2019

Pull requests may not seem to fit into agile software development, but they can work well if done right. They are problematic if they slow down integration, but if you can maintain feedback on your working software from frequent integration, using PRs to help people understand your code and review it can improve quality and reduce knowledge silos.

The speed at which PRs can be reviewed and merged depends on at least three factors: context, size, and atomicity.

Context is how familiar the reviewers are with the code and the business requirements. The more you know about the code base and the goal behind a change, the more you will focus your comments on significant issues rather than on either minutiae.

Including reviewers who have touched related parts of the code is one way to get code familiarity. Having pairs working on a story will guarantee that there is someone who can be a sounding board for design discussions as well as the point person on code reviews. Having a review partner can also extend common understudying of the code and reduce knowledge silos.

Size is the number of lines of code in the PR. The more changes, the longer it takes to examine and comment on the code. Breaking tasks into smaller PR units will both allow for faster feedback and encourage thinking in terms of smaller, self-contained units of work that can be more easily understood.

Atomicity means having the PR do one thing. This is all about how you structure your work: Breaking work down into small steps will make it easier to create smaller PRs, and focusing on one aspect of functionality will make it easier for reviewers to focus comments.

Atomicity is related to context and size. Atomic PRs are usually smaller: The more atomic the purpose of the PR, the less likely a developer will be to add increments of additional work and expand the scope of the review.

While context, size, and atomicity all describe the structure of the changes, they also have an impact on the mindset of the reviewer. The more daunting a task is, the more likely we are to defer it until we have more energy; someone reviewing a five-hundred-line change about a feature they are unfamiliar with is more likely to push it off, whereas a small change about something one understands well feels easier to address immediately. Delays due to waiting feel more significant (and avoidable) than delays due to work in process time.

While it's not always possible to make changes small and atomic and to have reviews with the right context, aiming to do so will make for faster pull requests and better shared understanding of the code.

If you implement pull requests within the spirit of agile feedback mechanisms, such as pair programming, you can avoid the downsides of branching and pull request reviews while obtaining the advantages of shared understanding and additional feedback.