techbeacon.com

6 ways QA can work better with developers

Andrej Skeledžija

As a QA engineer, I know how easy it is to get frustrated with developers. Often, these frustrations are the result of differing priorities: Developers focus on delivering code, while QA cares about quality.

Agile teams can help QA and developers align, but even then there are many pitfalls to avoid. In my experience, there are six key best practices QA teams should adopt to improve collaboration with developers. But first, let's examine why the teams come into conflict in the first place.

When is "done" really done?

The core difference lies in the motivations driving each team. Developers are often assigned to write code that addresses a specific function. This technical, limited scope can cause them to miss how their piece fits into the bigger picture. QA engineers, on the other hand, are tasked with bringing a holistic, user perspective, in which the quality of a specific feature is only as good as the overall experience.

For example, I was once involved in building a system that monitored the number of active users. The way the algorithm was built, it reported back fractional users—say, 5.5 active users—which made absolutely no sense from a user perspective.

In another example, a developer proposed requiring users to manually delete items in several places. This required the user to perform multiple actions that could easily be automated, significantly affecting the user experience.

But let's be proactive: What can QA do, and what state of mind should it adopt in order to realize the potential of agile development? Below I've listed a number of ways testers can improve their working relationships with developers.

1. Focus on quality, not on testing

Testing is only a means to an end. Too often, QA engineers run tests as if they're expected to fill some kind of quota for completing tests. We must remember that the real goal is to improve the quality of the product.

Make sure you understand what is important for your customers and test that. Don't just look at the user story definition. Try to think like the users and make sure the application will make sense from their perspective.

In one case I was involved in, the application had a error-reporting feature that passed all the functional tests and had a great look and feel. But customers complained that they weren't able to easily understand from the report where the highlighted problem actually was.

Always think of your users and don't test just to say that you performed a test. Users don't care how many tests you ran on your application—they care about the product quality and that it answers their needs.

2. Share responsibility

It's very simple: Everyone should be responsible for the quality of the product. In an agile team, there's no more "us" and "them." Developers should be responsible for their code and make sure they write the relevant unit tests. QA should test the whole system.

Yes, QA are the gatekeepers, but everyone in the agile team should have the same responsibility for the quality of the product.

3. Choose your battles

As a gatekeeper, you can't fight over every single defect. Understand which fights are worth fighting and where you can let go. Otherwise, everyone will spend a lot of time fixing things that are not that important.

Here's a tip: Define your own "red line" of things you simply won't compromise on, and focus only on those things. For example, I am very particular about initial interactions with a product (e.g., load time, usability of initial flow, etc.). Conversely, I've avoided fights about installation tweaks for on-premises solutions that only get installed once.

Many teams set up a "defects committee," which predefines showstoppers vs. other bugs before each sprint or version is released. This helps focus everyone's efforts.

4. Be constructive about defects

No amount of testing will ensure that you have zero defects. Some will always escape even the most rigorous testing procedures and be discovered by external users. The key is to keep calm, learn from these "escaped defects," and improve your next release.

Developers love to ask QA engineers, "How did this get past you? Didn't you test?" The reality is that software gets very complex, and you can't always test every single scenario and configuration. We conduct risk-based testing and test the user flows we see as most important and common according to the time we have.

In some cases we consult with product management, commercial stakeholders (sales, pre-sales, etc.), or even the customers themselves. If something gets through our web, we do a debrief to discover what happened and why we missed it, and, we create an automatic test for escaped defects.

5. Create visibility into your activities

Visibility improves collaboration and trust in any team, and agile teams are no exception. You shouldn't assume that developers or any other stakeholders know what you're up to, how you're testing, or what your standards are. Review what you're planning to test with the developer.

When you share your work with developers, they can start paying attention to the things that matter, upstream. Having bug-hunt days, with additional stakeholders from product management, support, and architects, not only widens the scope of testing efficiently, but also gets more eyes to scrutinize the product. Publishing important lessons learned from customers has the added benefit of establishing you as a subject matter expert who represents the end user.

6. Don't punish developers or users

I often hear QA threaten that they won't approve a feature because of low quality. In my opinion, this is really the worst thing you can do as a tester. Think of the outcome here: By not approving a feature, you alienate the developer, and, worse, your users will not have the chance to use it.

There are many things you can do in case of low quality: Have a task team dedicated to improving quality; only release the first part of a feature (one that is sufficient quality); and the list goes on. A common tactic is to mark a feature as "alpha," "beta," or "early access" to set expectations. This means that users will be able to start using the new feature, understanding that it's perhaps half-baked. I think this is a win-win: The users get the feature, we get feedback from them, and our "quality conscience" stays intact.

Take initiative

Skilled QA engineers don't wait for some magic process to make everything run smoother. Instead, they take initiative and improve collaboration with their developers, remembering above all that delivering a high-quality experience to the user is paramount. Testing is a means to end—that end being the quality itself.