I solved Scrum testing bottleneck problem!

S Samer Salib Follow
Jul 24 · 5 min read ★

Does your team suffer from most stories getting closed near sprint-end and QA testers struggling to keep the demand of closing the stories? Do your developers get QA testing feedback last minute and their stories carry over to next sprint? Read on!

© Matt Kenyon

Why should you read a Scrum solution article from a mere developer?

I'm that one developer who moved from having frequent one-on-one conversations with his manager about stories carrying over to next sprint to becoming the developer who is always impressing the team by how much work-ahead he has completed from future sprints. Do I have your curiousity peaked now?

Problem to fix:

All developers deploy code that requires QA testing near the end of agile scrum sptint. Good job! They are even a day or two early! Yay! Now QA team members, which seem to have always been outnumbered, brace themselves for the storm, and they start testing this many features in the span of a day or two. QA team members step to the plate and they make it happen! Marvelous 👏👏
Now developers get feedback on what they missed (a topic for a new article) and now they have to fix it in a day or two before sprint review day, which increases intensity and stress for EVERYONE!

What is not a solution!

- No, having more QA team members won't fix it.
- No, developers should not wear a QA hat to meet demands, although dev testing and code review testing is very important to reduce QA feedback cycle.
- No, breaking stories to smaller stories doesn't change the fact that at least one of these stories will be completed near the end of the second half of sprint.
- No, having one-on-one conversations with developers to learn the challenges which lead to their stories carrying over to next sprint

won't fix your broken agile system. Sorry, I'm getting flashbacks.

- **And No, switching to Kanban or a hybrid agile approach isn't the ultimate solution! Why? Because no system is perfect. You don't know the limitations of a system till you start to adopt it.**

So, what then! Is Scrum just a bad system that has necessary evils to live with!?
No, Scrum has been successfully used for years and **works very well** for those who **know how to use it**. Just don't misuse the system and wonder why it's not producing **positive results**. You will know what I mean when you **learn the root cause** of this problem, next.

The root cause:

It took me a while to figure this out, even after practically implementing the solution shared in this article to beat Agile woes for months so successfully!

I learned the root cause when a diligent scrum coach substituted for our scrum master and **she in her Agile wisdom, enforced an Agile rule that is so ignored in the fast-based projects ran by semi-professional scrum masters**, which is to **give developers story points in the sprint that will get done in LESS THAN sprint cycle days number**. Read it again. To put this in perspective, in my team's two-weeks sprint, 13 points is the max and stretch of load to each developer. In my team a 13 point story in complexity means that developers think it takes 7 days to get done. That's right 7 days! Not the full two weeks. And don't worry, your developers won't be spending the rest of sprint days reading Medium articles! They will be… oh wait, that's actually the solution! Here it goes, next.

Solution!

Two words…
>>>> ***Work ahead!*** **<<<<**

The solution is for developers to work ahead on items from next sprint. Which will result in those work-ahead items being ready for testing:

1. Before the sprint starts! Great, but current sprint stories are high priority to test!
2. Early when the sprint starts! QA will have work to test in the first week!
3. At the very least mid-sprint or shortly after! A major improvement over finishing a day or two before sprint reivew.

This gives developers time to respond to testing feedback and complete their stories within the span of sprint, with minimal risk or unnecessary stress.

But how can they start on items from next sprint if they have so much to finish in the current sprint!?

Let's see…

Well, first of all, don't give them "so much to finish" in the current sprint, and watch them surprise you with how much they will deliver over all.

In all three cases of work ahead results mentioned above, when all sprint stories are deployed early on, developers will surely almost always have plenty of "free" time to work ahead on more items from the following sprint, which is a proactive approach that will improve developers productivity and well being. The team overall dynamic will shift to positive vibes compared to the vicious stressful cycle of wrongfully implemented "Agile" full-capacity that actually defies the proper Agile rules of under-estimating and taking no more than what can be done WHILE leaving time for testing.

Once you make the right decision taking on less than can be done in two weeks days number (or whatever your sprint duration is), mostly whenever possible, you will find developers more productive, and multiple stories completing within the sprint even from future sprints! Also, QA testing work-load will become better balanced throughout the sprint.

Work, work, work! It will be done either way! Time spent and efforts made. Best to enjoy it while getting it done, than to get it done with stress and dread.

Made it this far!

Next will be an article on how improving communication between developers, other developres in design discussions, product owners and business analysts who draft user requirements and QA team members about functionality can reduce over-all story completion time by reducing PR comments and potential reworks and by reducing functionality assumptions and miscommunications that usaully turn into “defects” or rework based on testing feedback. Stay tuned.

Happy Coding Everyone!

Agile Agile Testing Agile Project Management Agile Scrum Methodology Scrum Master