Improving Cross Functional Collaboration

Dain Peter Charbonneau

dcharbonneau@paraport.com

Abstract

Testing should not be an island and we should not be throwing code back and forth over a wall, however this is what happens when there is poor or no cross functional collaboration. Cross functional collaboration is essential for successful agile environments.

Everyone has opportunities to encourage and implement behaviors that improve Cross functional collaboration. This also allows additional benefits such as getting out ahead of defects by driving quality upstream, encouraging earlier QA engagement, greater understanding of the product and user behaviors, and an improved ability to identify non-functional defects.

This paper examines practices that can be used for improving cross functional collaboration, adding to this are the authors experience with this practice, pitfalls encountered, and successes.

Biography

Dain Charbonneau has 11 years of quality assurance experience working in ecommerce and the financial service industry. Throughout his career, Dain has been a mentor and coach, worked on enriching sites through A/B and multivariate testing, led QA teams on feature development for large and small projects, and is currently working on an Agile team with a DevOps focus. Contact Dain at dcharbonneau@paraport.com.

1 Introduction

Cross functional collaboration involves people with different roles and expertise contributing to a common project or goal. These people may be grouped into a specific team for the duration of a task, project, or longer term for a company strategy or initiative. Additionally this involves people not dedicated to a specific team whose intentions are divided on many differ tasks and projects.

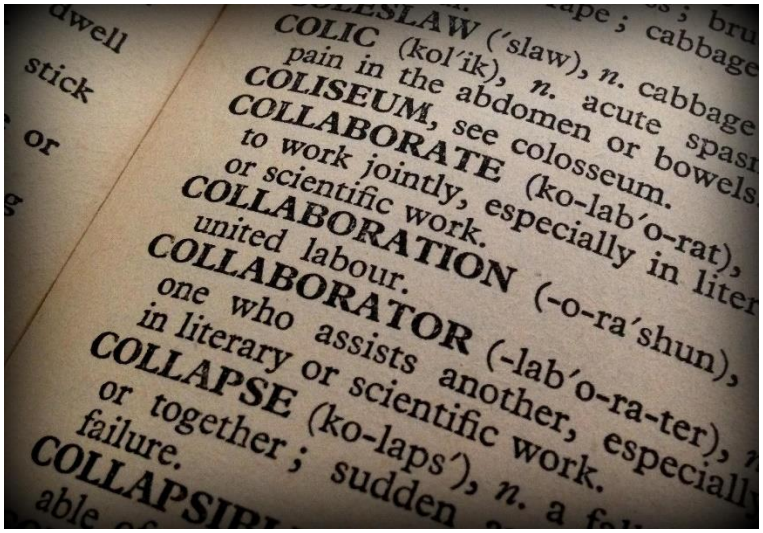

Image 1 (Pixabay, n.d.)

There are a number of benefits to a work culture that encourages and supports cross functional collaboration. A few of these benefits are:

- Improved project visibility across departments. This helps avoid gaps with what's delivered vs expectations.
- Identify misalignment of project goals earlier in the projects life cycle. This can reduce late scrambles adding missed features or pushing out features to future projects.
- Helps to improve the entire teams understanding of the feature and can aid in identifying bugs earlier.
- Gives people a stake in the project outside of their specific area.

An individual cannot always dictate exactly how a team functions, let alone additional teams one may have to work with due to common tasks and goals. But that does not mean an individual has no power to encourage cross functional collaboration. There are many opportunities for professions to incorporate practices that can develop, improve, and strengthen cross functional collaboration throughout the software development lifecycle.

2 Why poor cross functional collaboration is a problem.

A key component of Agile development is cross functional collaboration. A team can claim to be Agile, but with poor cross functional collaboration they are effectively tossing handoffs over walls to each other. Poor cross functional collaboration is a leading contributor to project delays and failures.

What is the problem?

- Poor cross functional collaboration negatively affects performance. (Schenk, 2016)
- A lack of collaboration is cited as a leading cause of workplace failures from a survey of 1400 corporate executives, employees, and educators. “86 percent cite lack of collaboration or ineffective communication for work place failures.” (Stein, 2012)
- Cross functional collaboration is challenging when:
    - People are only partially dedicated to the project.
    - Team members are in different locations around the globe.
    - Poor tools and/or processes in place to support Cross functional collaboration.
- Moves teams closer to separate entities where code is tossed over the wall for each step of the process.
- Individuals don't feel they can implement change. Instead complain hoping and waiting for leadership to fix everything.

The blueprint for building an agile team with a strong culture of cross functional collaboration can be found in many books and all over the web. Examples are Martlew’s book Changing the Mind of the Organization: Building Agile Teams, (Martlew, 2015) Adkins book Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition, (Adkins, 2010) and web sites like Atlassian’s Agile Coach. (Atlassian, 2018) Just because a team is functioning exactly to an agile blueprint does not mean that there are no opportunities for improvement. Teams are made up of unique individuals, no two teams will ever be the same and no blueprint will work perfectly for every team. A blueprint for agile can be an excellent guideline for implementation, but to excel a team should be adaptable, willing to try different things, and always looking for ways to improve.

3 Techniques to improve cross functional collaboration.

Communication is key to successful cross functional collaboration. There are a number of practices that can help to encourage and improve communication across departments such as development, quality assurance, business stakeholders, project managers, and customers. Not all practices require a mandate from high up or a major overhaul of a work culture. Many practices can be introduced and encouraged on an individual level.

3.1 Developer handoff demo.

A powerful tool that can be used to ensure a feature is really ready for testing and ensures the testing professional understand the feature is for the developer to demo their code. This involves having the developer explain their code to the testers and display how it works. This can easily be initiated and driven by the developer or testing professional and does not have to be limited to these two professions.

3.1.1 Why use the developer handoff demo?

The first benefit lies in the description, the developer has to show it working. There are many stories of experiences where a handoff should have never happened, the code was just not ready. This can waste countless hours troubleshooting, debugging, and filing bug after bug without the ability to fully test the feature. Developers may then waste additional time getting refreshed on the code to work on it again. Requesting a demo of the code at hand off can save time for everyone involved in this stage. Think of the time and effort saved if during the demo, it’s identified that a piece of the feature is not working. For

Image 2 (PXhere, 2018)

quality assurance, no time is wasted testing a feature that is not ready. As for the developer, the code is fresh, resulting in quicker turnaround time due to the reduced ramp up needed when reviewing to make corrections.

Another benefit a dev demo has is how they allow the testing professional to review the code with its creator. This is an opportunity to clear up any uncertainties the tester may have. This is an opportunity to ask questions while the code is fresh in the developers mind. The tester should walk away from the demo with a good understanding of the item being handed off, how it works, and how to test the feature.

Finally, this is beneficial to the tester as they gain insight into how the developer uses the feature. Understanding how the developer uses a feature can help to identify areas overlooked with development and target test cases that are not covered with unit and integration tests. This can aid in setting priorities in testing to identify bugs quicker.

3.1.2 Only when its needed.

A developer handoff demo should not be requested for everything as they take time and add meetings to the schedule of those involved. Questions should be asked prior to requesting one to ensure the dev handoff demo is being used effectively: Is the code extremely complex? Is there confusion with some portion of it? Does the tester understand the feature? A no to any of these questions is a good indication that a code handoff demo can be useful. Be smart in knowing when a demo can be of benefit and use this tool effectively.

3.1.3 How its used in the office.

In my current workplace this is encouraged by leadership whenever a tester needs to improve their understanding of the feature being handed off. If there is confusion with the feature, we request a demo from the developer. A request is not always made to initiate a demo, developers will also choose to demo their feature if they feel it's needed with the feature being handed off. Most of the time I find it sufficient to do the demos while pairing at a desk, only requesting a conference room as needed for larger groups.

3.2 Test case reviews.

Another technique that encourages collaboration between testers, developers, and feature experts are test case reviews. This is where the tester walks through the test cases with the parties responsible for the code and any feature experts.

3.2.1 Why have test case reviews.

Developers should know their code better than anyone else, they wrote it. There are likely product owners or business associates that know the features expected behaviors better than anyone. Fellow testers will have insights based on their experiences and different understandings. A test case review provides opportunity for attendees to bring insights with testing and additional risks that tester was unaware of or was overlooking.

3.2.2 Key for effective test case review.

Attendee engagement is key for this to be successfully. Be smart with who is being invited to attend the review. Include those that are willing to speak up and help generate ideas no matter how obscure an idea may be. If all attendees think the test plan looks good, challenge them to come up with ideas even if there are low priority edge cases that may never get tested. Try to keep these review groups small with involved players as this helps to avoid ideas being drowned out by stronger personalities.

3.2.3 Benefits of test case reviews.

Test case reviews provide attendees with insights on how and to what extent QA is testing. Because they are being asked for feedback on the test methods and coverage, attendees can get a sense of having a stake in the testing. This is important for the developer and tester relationship because they are most effective as a team working to make the code better vs the destructive environment of us vs them.

3.2.4 Test case reviews in practice.

For my current team, we strive to have test case reviews for most projects. My team has been fantastic in these reviews with calling out additional tests or questing existing ones. These reviews can be difficult for the testers with test cases being challenged and called out. We need to understand this is an opportunity to learn and improve rather than being defensive. The result is improved testing coverage, improved understanding of the feature and project, and allows everyone to learn and grow from each experience.

3.3 Daily stand-ups can be effective even if they do not fit the mold.

A very common agile practice are daily stand-ups. Daily stand-ups are short regular meetings that are part of agile methodologies. The intent is to have everyone standing so people do not get comfortable and to keep the meeting short, preferably under 15 minutes. The meeting consists of everyone answering three questions: What did you do yesterday? What do you plan to do today? And do you have any roadblocks or impediments?

When executed well, these can be a great resource for collaboration on your team. They provide insight and understanding towards the big picture and allow for visibility on blocking issues. When not utilized or executed poorly, an individual's focus can become tunneled to their current work which can result in missing the overall picture. The stand-up is an opportunity to gain awareness of what the team is doing as a whole, bring to light issues, and set up collaboration among the attendees.

Typically, when implemented, stand-ups follow agile best practices guidelines. However, for stand-ups to have long lasting effectiveness, teams should constantly evaluate what's working, be willing to try new things, and make adjustments for areas that are identified as broken.

3.3.1 Keeping stand-ups small.

What happens when a team or project grows too large, to the point where stand-ups are excessively long losing their effectiveness? Too many involved they will become disengage and feel that their time would be better spent elsewhere. One option is to consider shrinking the stand-up back down by splitting up into to smaller groups. Each group would then have a designated spokesperson that would meet with the designated people from the other groups to relay any information pertinent to all. This process is known as a Scrum of Scrums. This does not mean bringing more people together is always wrong, just be sure the situation calls for a larger stand-up and try to avoid making a habit of it. It's suggested for when groups get very large to no longer call the meeting a stand-up, but to rebrand it to help set expectations that differ from a stand-up for the meeting.

3.3.2 Asynchronous stand-ups.

Another option to managing large stand-ups and involving global remote attendees are asynchronous stand-ups. These are stand-ups that are typically managed through a chat bot. The bot will ask the three standard stand-up questions, or more if defaults are changed, and at a designated time will share all answers through the chat. An example of a bot for asynchronous stand-ups is the Standup Alice bot in Slack. Image 3 shows the slack bots initial greeting to start the gathering of information.

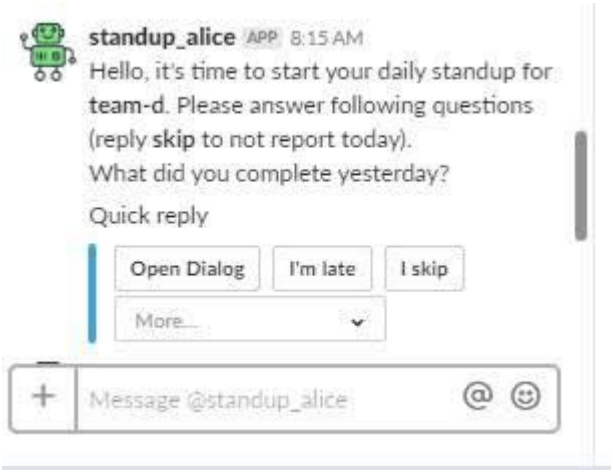

Image 3 Standup_Alice bot in slack to support asynchronous stand-ups.

During large team stand-ups, people will lose focus and may miss key points. But with asynchronous stand-ups there is a record saved so it can always be referenced. This is also beneficial to those who are unable to attend the stand-up but have an interest in it. Conflicting meetings are not a problem as the stand-up content can be reviewed as availability allows. This is also convenient for remote team members, attendees around the globe do not need to join stand-ups at weird hours due to different time zones.

There are risks to be aware of with asynchronous stand-ups. Blocking issues may not be addressed right away, since asynchronous stand-ups do not always happen at the same time for everyone. Since brainstorming is not done face to fact, they may be a reduction in ideas generated. There is also the loss on interpersonal team building that occurs from coming together in person.

3.3.3 Hybrid stand-up.

To combine the benefits of a traditional stand with the advantages of an asynchronous stand-up, consider a hybrid stand-up. This starts with an asynchronous stand-up, after which everyone still comes together in person or video conference for remote attendees. This provides an opportunity to address questions or concerns from the asynchronous stand-up. This also keeps the face to face interaction short as the three questions have already been answered and reviewed by everyone. Only clarification questions and issues are discussed. For those who are unable to attend and interested, there is still the record for review.

3.3.4 Personal experience with stand-ups.

Having experienced recurring oversized stand-ups where attendees would become disengaged and discussions were not relevant to most in attendance, I have a strong preference for scrum of scrums and asynchronous meetings. With my last experience of an extremely large stand-up, we had good intentions with different teams forming a joint stand-up for a specific project. However, the net result was the majority of people's time being wasted. Once it was acknowledged that this was not an effective use of our time, we broke back up into smaller stand-ups with pertinent information being shared by the leads. Currently my team is executing the hybrid stand-up. When we do this right, it has helped to keep everyone informed and engage with our colleagues in Argentina. This also keeps the face to face time short and effective. When done wrong, the traditional strand up takes place after the asynchronous and the effectiveness is lost. We continue to work on and improve our implementation of this.

3.4 Engage stakeholders and users.

There can be disconnects between stakeholder expectations and what is being delivered. Perhaps there was a misunderstanding, perhaps something was missed or left off the acceptance criteria, or perhaps it was unreasonable with the time and resources allocated. But the result is the same, when delivery arrives, the stakeholders are disappointed. One way to help avoided this is by engaging stakeholder and internal users throughout the development process with demos.

3.4.1 Demo to stakeholders and users.

Providing visibility through the development of a project to users and stakeholders can provide a number of benefits. Demos help to ensure everyone is on the same page with deliverables. There are no surprises at the end. If there are concerns with anything, these concerns come out much sooner in the product life cycle. This feedback loop helps to establish and manage expectations, identify disconnects, and if changes are needed due to disconnects, development does not waste time ramping back up as they are currently engaged

Image 4 (PXhere, n.d.)

What if there is nothing to demo but still want to build engagement with the project?

- Share metrics.
- Talk about the benefits the feature will provide.

Attendees are taking time away from their work to come see the demo. It's important to keep demos on time, focused, and to the point. The goal is not to bore everyone, you want to engage attendees, build excitement for the project, and invite involvement and engagement.

3.4.2 Situational settings for the demo.

My office uses different techniques for this demo depending on situations. There have been quick demos at the end of stand-ups with stakeholders attending. We pair at a desk for short demos to an individual stakeholder. Last, conference rooms are used for short demos as part of a meeting or more formal demos that are the purpose of the meeting.

3.5 Pairing with User Acceptance Testing (UAT) & Test in production with partner users.

When handing off for User Acceptance testing there is the old throw it over the wall methodology. Evolving past this old methodology, there are better ways to hand off for UAT.

3.5.1 Demo the UAT handoff.

Similar to how the developer demos a feature to the tester, the tester can demo the feature to those conducting UAT. First demonstrating what the user needs to do to access the feature. For example, if the feature is in a test environment, the first part of the demo will be how to access the environment. The next part of the demo is showing the feature. This helps prevent confusion with how to access the update and establishes exactly what was included with the feature.

3.5.2 Pairing during UAT and with partner users.

The second option that can be used is pairing during UAT. This is where the testers sits down and watch the users use the feature. Done right and improved understanding can be gained on how the product is used by the user. If a bug is encountered, the tester will have a clear understanding of it rather than getting a vague email and trying to reproduce it. Last, this provides opportunity for quick feedback and discussion if there are concerns. Below are suggested tips for effective pairing.

- Don't influence the experience by showing or explaining expectations.
- Have the stakeholder use the product without giving steps. Provide them with a task and see how the task is completed. Many times, a behavior outside of expectations will be observed. If the user can not figure out how to complete the task on their own. Then there may be a UX design flaw.
- Ask about scenarios that are not used. Are there better ways or does the user not know about capabilities.
- Conducting multiple sessions with different users, especially if there are multiple groups that use the feature. Do different groups use the different ways?
- Smaller "focus groups" with 2 to 3 people max. Larger groups will drown people out, especially with direct reports in the room. People seem to be less vocal on opinions when their boss is present.
- Have multiple sets of keyboards and mice set up. This encourages getting others involved rather than only one person driving with the mouse and keyboard.
- Consider inviting a developer or two to attend. Developers can also learn for observing how the feature is used in the real world.
- Prepare what if questions. E.g. If a feature breaks what do you do? If someone asks you to do something, how do you do it? If you could improve something about this flow, what would it be?

Pairing can be an effective exercise and should not be limited to the UAT process. Depending on the workplace and industry this exercise can be an effective tool for testing in production with partner users.

3.5.3 Pairing during UAT and with partner users personal experience.

Depending on who the user is, many companies need to set up focus groups to observe their users with new features. It's much easier with my company, all that is needed is to set up a meeting. This is due to the end users being the portfolio managers and traders in the building. It has been an effective tool in improving developments understanding on how products and features are being used and in creating effective solutions.

3.6 Building work relationships.

3.6.1 Advantages of building working relationships.

Positive working relationships allow employees to be more comfortable in interactions with colleagues. This can be extremely important when collaborating with people across different teams and departments. Getting to know coworkers such as subject matter experts and developing working relationships only

pays dividends in the future. Take advantage of opportunities during and outside of work to get to know colleagues and develop working relationships. Establishing a positive working relationship will encourage behaviors like collaboration and team centric thinking.

The challenges are amplified when attempting to improve cross functional collaboration with colleagues in different offices or regions. If possible put in the extra effort for face to face interactions when the opportunity to meet in person exists. Set up team activities outside of work to build working relationships among different offices and regions.

3.6.2 Personal experience example.

When a new colleague visited from Argentina, the team made it a point to do outside work activities. While not everyone attended all the team lunches, happy hour, and dinner that was set up, everyone did make it a point to make it to multiple engagements. This was done to help establish interpersonal relationships with the team's new colleague and to create building blocks for communication and collaboration in the future.

4 Conclusion

Cross functional collaboration is an important component for productive work environments, successful projects, and effective teams. Without effective collaboration, there is a shift from agile to a silo style development and the us vs them mentality. This can lead to longer delays, roadblocks, incomplete projects, and increased hostility in the workplace. A culture of cross functional collaboration can combat these issues and lead to improvements in projects and their development life cycle. Building working relationships across functions with face to face interactions, providing information and feedback between cross functional roles, and having bigger picture engagement all contribute to a culture of collaboration. There are many opportunities to build engagement between departments and employees in different functions through the development life cycle. Taking advantage of opportunities and molding them to best fit your people and company will contribute in creating a more enjoyable and collaborative work environment.

References

White, Stephen. 2016. “Why Your Team Should Try Asynchronous Daily Stand-ups” Medium, entry posted November 4, https://medium.com/@stevoscript/why-your-team-should-try-asynchronous-daily-stand-ups-87f1b809e5c8 (accessed July 16, 2018).

Appelo, Jurgen. “Daily Meetings with Remote Teams (Stand-ups Don’t Work, But Daily Cafes Do)” Agility Scales Blog, entry posted July 4, 2017, https://blog.agilityscales.com/daily-meetings-with-remote-teams-stand-ups-dont-work-but-daily-cafes-do-35c6d3902f3b (accessed July 16, 2018).

Stein, Nick. “Is Poor Collaboration Killing Your Company? [Infographic]” SalesForce Blog, entry posted Sep 12, 2012, https://blog.agilityscales.com/daily-meetings-with-remote-teams-stand-ups-dont-work-but-daily-cafes-do-35c6d3902f3b (accessed August 18, 2018).

Schenk, Tamara. “How Cross-Functional Collaboration Impacts Performance” CSO Insights, entry posted Oct 20, 2016, https://www.csoinsights.com/blog/cross-functional-collaboration-impacts-performance/ (accessed August 18, 2018).

Atlassian. “The Agile Coach: Atlassian’s no-nonsense guide to agile development.”Atlassian Agile Coach, https://www.atlassian.com/agile (accessed August 18, 2018).

Adkins, Lyssa. 2010. Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition. Addison-Wesley Professional.

Martlew, Christopher. 2015. Changing the Mind of the Organization: Building Agile Teams. Troubador Publishing Ltd

Pixabay (n.d.). Dictionary Book Collaborator Words. [image] Available at: https://cdn.pixabay.com/photo/2015/12/23/22/16/collaboration-1106196_960_720.jpg [Accessed 25 Aug. 2018].

PXhere (n.d.). Collaborative meeting conversation. [image] Available at: https://get.pxhere.com/photo/meeting-conversation-convention-academic-conference-205854.jpg [Accessed 25 Aug. 2018].

Pxhere (2018). Coworkers in a business discussion. [image] Available at: https://pxhere.com/en/photo/1434347 [Accessed 25 Aug. 2018].