The story I would like to tell is about a process change that I made early in my career as a QA Team Lead, about 6 years ago. My team was responsible for testing most of the client facing front end applications including the client onboarding experience. As far as a testing organization goes, we were relatively immature. The team was mostly made up of people who came from the business side of the company, with no outside testing experience. We used our business and corporate knowledge, combined with grit and determination to get the job done, but did not have exposure to practices that come with years of experience. The majority of the time, we were successful in our efforts. There would be issues that came up, but nothing disruptive to the business, until the project I am going to tell you about today.

The focus of the project we were working on was to create an experience that would make it easier for the client to upload the tax documentation necessary to open an account. Client feedback showed a high dropoff rate at this stage of the application, as it was confusing and cumbersome process. The project to improve this journey was greenlit to create a new service providing the client with a seamless way to select and add the proper documentation based off the data they entered on their application. Aside from this new service, changes were made to our existing account onboarding stack, including the online account application and the application that our internal team used to open accounts.

Testing this project required the testing team to understand what data drives the presentation of the tax document to the client. This is where the focus was. Unfortunately, there was some refactoring to the application that opened the client accounts, but this was not made known to the testing team. When the project was released, the client experience and the goal of the project was achieved, to make the client’s ability to select and upload the tax form simple. The problem was the application used to open the account had many major bugs. We were able to capture the data with the uploaded forms, but in many instances, were unable to open the accounts. This was a serious disruption to the business. As a unit, Development and QA were able to put patches in place to correct the issue. Everyone involved, and some of those not directly involved, worked as a team over a couple of long days to restore the account opening service. While externally it was not a high spot, the accountability and loyalty to coworkers, the company, and the job was something to be proud of. The teams dedicated themselves to righting the ship and putting the problem behind us without any finger pointing as there were wrongs committed by everyone involved.

As the lead of the team responsible for testing this release, I knew I needed to own the problem. After talking with the tester responsible for the application to understand how the release went wrong, I realized that communication was the heart of the issue. There was a breakdown in communication between QA and development and a breakdown in communication between QA and the business during the User Acceptance Testing (UAT) process. There was also a failure of communication internally within the testing team. I realized that if we wanted to be successful as a team, we needed stronger communication. This meant more frequent project checkpoints and increased transparency on the progress of testing.

I was called into a meeting with my boss who wanted an explanation. He wanted to know who tested this application and why we were in this position. My response was that while I did not test this particular piece of the project, I was responsible. It was my fault the scope of the change to the application was not known, it was my fault the handover from Development to QA was not done in a way that we as a testing team understood everything involved, and it was my responsibility to make sure we as a team do not repeat the same mistakes. The important part is that we learn from our mistakes and failures, getting better from them. What followed was a change in the handover process from team to team, including a more comprehensive UAT process.

First I addressed the communication with the Development team. Not only did I need to answer to my boss, I also needed to answer to the Development head, with whom I had a good relationship since we both wanted the same thing, successful seamless releases. After we understood the problem, we got to work on establishing an environment where communication between Development and QA was fostered and encouraged. We would have sessions in place to allow collaborative handovers with question/answer sessions so that all teams involved would understand the projects and application changes involved. This not only helped on a project-by-project basis, but built a community between the teams. We grew together as a unit.

Next was time to work on the UAT process. Previously the QA team would often handover the changes to the business unit to review via email. An email would be sent to a lead in the business unit explaining the change and asking them to review in the test environment. The responsibility was in the hands of the business to understand and work through everything. A change was needed to link the QA team to the business team as a single unit. At the time, we did not have established Product teams with product owners. To establish this link to the business units, I could not have a conversation with a single point person. Instead, my team needed to reach out to the respective business person who requested the change and properly handover the change to give them the opportunity to review and test the change. There was a shift in responsibility, it was up to my team to make sure the handover was properly. My emphasis was to treat the handover as if you were the one receiving it. I worked with my team to formulate handovers with testing documentation, test data, and lists of observations noted during testing. This allowed the opportunity for questions, answers, and collaboration between the units. We created an environment where the business and testing units were free to ask questions and challenge each other. Better for these challenges to happen in the preproduction environment as opposed to production. It took some time to get the process flowing, but once the units were in sync, the number of issues impacting the production environment were reduced.

Just because we were seeing success, it did not mean that the learning and adaptation of our work stopped. The processes we followed were flexible enough to allow modifications based on the context of the release. The processes were guidelines, not strict rules. If there was a better or faster way to accomplish a given step, it could be followed by the team, resulting in a living process, with the goal of always getting better.

As we matured more as a company, adding true product teams with dedicated product owners, we took our existing process and grew it even more. This evolved into a formal process that was integrated into our release workflow in JIRA. It put us in a position where the Development, QA, and Product teams were all working as one.

How can you tell if a change was positive or negative? Often times, it is difficult to tell the lasting impact of a process change. The may be an immediate impact or it may be a long term change where the impact takes time to develop. In this case, the results were seen by increased communication amongst teams, growing confidence in the process which allowed us to move more aggressively, and increased ownership by the teams. Recently we restructured our organization. A product owner who is now working with another Development/QA team remarked to one of my team members how she misses working with our team, with the way we worked together and handover releases. My response was that she needs to own her process and work with the teams to establish their relationship. They need to get to that same level of communication and trust that she had with our team.