Better Tester and Developer Collaboration

Video Transcript

Welcome to Original Software's demonstration series: Solutions in action. This module will review and demonstrate Original software's solutions with a focus on tester and developer collaboration. Sometimes the relationship between testers and developers can be less than ideal, although both roles are motivated to deliver high quality software systems that meet the needs of the business, oftentimes the effective collaboration between these roles are much less than optimal.

An example of this is a tester is given some new software to test and they really don't know what the software is supposed to be doing and there are insufficient or inaccurate requirements documentation. On the flip side, the developers are given less than ideal descriptions of the issues that testers report, this leads to the common phrase: 'Bug- well it works for me just fine, maybe she'll reproduce the scenario or better yet come over here and show me'.

Let's explore the first scenario a bit. A developer adds or changes some functionality due to tight timelines or agile methodology where requirements documentation is not fleshed out formally in any detail. The developer will indicate the system is ready for the tester to validate so the tester is left to try and figure out what the change does, maybe write up test cases, do the testing, but isn't really sure if it's right or wrong. Ultimately some feedback is given to the developer. This is inefficient and frustrating to both the tester and the developer.

So, let's see how we can improve this situation. After the developer finishes his work and has tested it in his environment, he can very quickly document the added or changed functionality and communicate it to the tester.

The developer simply uses TestDrive-Assist to record the relevant bits, TestDrive-Assist will attach to and start recording the application. The developer steps through the application where the changes were made. TestDrive-Assist will record all the screens and all the actions performed on the user interface of the application. This takes very little time, but will provide an excellent step by step scenario for the tester to follow later.

As was shown in the previous modules of this demonstration series, mark-ups can be made at any point during the recording. This time we will do the mark-ups after we have finished recording. The developer can quickly highlight what was done in the context of how to navigate to his portion of the application. This kind of information is invaluable to the tester. After the developer has finished marking up the recorded result it gets saved to the repository and a new test case and task can be created which makes finding this recording easy. It is also a great way

to communicate to the tester that something new is ready for them to take a look at. The developer simply assigns a new task to the tester. The recorded result with the developer's mark-up will be attached to the task and the test case kept for future use by the tester moving forward. Now when the tester starts to work on this new or changed feature, they know exactly what was done by the developer and what they should be looking for. If the tester would like to quickly visualise the steps carried out by the developer and review any mark-ups just animate the results.

Now for the second scenario, the tester is validating the software system under test and finds a bug or issue. Maybe this is a hard to reproduce issue, maybe the tester describes the issue too loosely. In either case the information collected and presented to the developer to investigate is limited, then the developer will spend time trying to understand what has happened. They are challenged with being able to reproduce it and ultimately, they reject the bug out of hand or if they are nice, they may ask the tester to clarify or they may say: 'come over and show me'. Needless to say, this is very inefficient, frustrating to both tester and the developer.

So again, TestDrive to the rescue, the tester executes the manual test and records it with TestDrive-Assist. Even if the issue is hard to reproduce or maybe the tester is doing exploratory testing just trying to break things, the key here is the tester only needs to identify an issue once, the recording made by TestDrive-Assist will capture all the details of how to reproduce the issue.

Here is where you change the TestDrive-Assist view to show what was recorded instead of the test steps checklist. When issues are found just double click on the screen that is captured to display it, then mark the screen to highlight the issue and add a comment. During the process of saving the recorded test result to the repository, the tester can raise a defect and the recorded result will be attached to it.

Now the developer will see a new defect that has been raised and is assigned to them, the defect record is opened, and it's attached recording is open to see the details and the full context of how to reproduce the issue. The developer can now see all the screens leading up to the issue and can drill down into any screen that have mark-ups. In fact the developer can take this recording and quickly turn it into a TestDrive script.

Using this script they can play back the tester's actions against the application to reproduce the issue themselves automatically. The developer can examine the results of the playback and see if the application did indeed misbehave as it did for the tester comparing what was expected, which was the tester's scenario, and the actual which is what the developer sees now. Now that the developer knows exactly what the issue is remediation happens much sooner without the frustration inefficient miscommunications they suffered from before.

Since Qualify supports customisable workflow the developer can progress the status of the defect to a fixed status and assign the defect back to the tester.

To summarise, using Qualify and TestDrive-Assist, tester and developer collaboration is enhanced greatly. You will benefit from much faster mediation of issues and much better teamwork. The developer sees what happened and can even use the tester's recording to reproduce the issue, then he reassigns the task up to the tester for confirmation of the fix.

Thank you for viewing this module there are several others to see so please do. If you have any questions simply get in touch with Original Software- original thinkers and innovators.


Original Software