Source material for training sessions for the week of 1/13 – 1/17

Software QA for Art1: Writing better test cases, Better Jira Ticket writing. Making Stories better, Having some teeth.

[[

There is one perspective that test cases need to consider a variety of conditions that the software will be expected to handle. I tend to operate from the single point of responsibility principle. Each test should test one thing. Now frequently we have multiple things that needs to be tested in a workflow but we need to break this down to the lowest level that we can. Where we must test multiples, we must report on multiples in order to say Passed.

But back to Single point of responsibility: There must be some logical framework that houses and catalogs your test cases.

When there are hundreds of testcases without a structure to define them, the bulk of it all will cause a collapse that you cannot recover from.

Test case growth. As a product matures and ages the number of the test cases increases. It takes longer to execute all the test cases. Time does not grow longer as products mature. In fact time is usually shortened because management thinks the maturity of the product equals less time necessary to test it.

It is at this time when intelligent testing needs to be implemented and Test Automation is absolutely necessary.

]]

Here is the progression of a feature

Mind of someone

Jira Story

* Specificaion
* Description
* Etc.

Software development

Test case development

Feature delivery

Test case execution

Pass

Or

Fail – resulting in Jira ticket

Where can we optimize effort

**Shift left**

On aspect of shifting left is to improve the features definition.

I know there's not enough time. But getting this skill and ability will reduce your effort in the long run.

Features that do not have provable effects cannot be tested. The test engineer must learn within a team or a company how to push back. The clear communication that must be brought to the teams and product owner when this occurs is to say, “ This cannot be tested with the information specified.”

There must be some way to repetitively execute the behavior of the

Work with developers. Do your homework about a feature. Know what to ask developers about a feature and how it can be proven to be working correctly. This might be evidenced on the update time stamp in a file. It might be an entry or update in a db table.

Testing

**Test process maturity**

As a product is first developed and tested testing cannot be completely exhaustive and comprehensive. The product will not be stable enough for that kind of full force testing that we could do. But once stability is in existence, that is what should be tested or features should be tested to that level.

As a project develops test cases must be exhaustive and comprehensive. There isn’t anything else an application and early testing can go.

Once the development work that has lead to the stability and strength of the app is complete this focus of testing no longer needs to be the daily focus. But the testing shouldn’t be put away and ignored. You just use it once a week now instead of once a day or hourly.

Now testing will focus at a feature / story level.

Test categorization and leveling and tagging.

Testing Roles

I think we do this fairly well on the Delta CX products. Applications are fairly limited to the roles that different users have when they login. So, we don’t have such distinct different functionality and workflows for different users. But we still need to consider and account for the effect that a different login has on an application. In fact there needs to be a test case collection that proves the functionality is no different regardless of user ( of a specific class / role of user )