**Brainstorming On Software Quality Assurance**

When we start thinking about quality and what’s best for our particular project, it’s a good idea to have an opinion as to what we plan on doing and why. Now this is a question that has varying opinions on what the right answer is throughout the industry. Some will argue that development should begin with writing tests, while others believe that it’s only necessary to begin testing once functionality is quite mature.

Regardless of when we believe it’s an appropriate time to begin formal testing, it’s always wise to include everyone responsible for quality as early as possible in our cycle. The more familiar these people are with what our goals are, the easier it will be for them to determine the best test strategy. Ideally, will be focused on testing early and often to identify bugs as soon as they arise. It’s important to keep in mind that the sooner we know about an issue, the easier it is to address.

Let’s take an example that takes a step away from software development entirely, but it still embraces some of the principles of incorporating testing early. Now, when we build a house, we need blueprints, materials, people to put it all together, and someone who will manage the overall project. Before we start, we generally apply for permits. And then we will have some inspections along the way. Let’s say we have our basic framing for our house and an electrician is starting to run wires that ultimately will be concealed within the walls.

And nothing else in the series of steps would have been affected. In actuality, we would have saved quite a bit of time and money. If we discover an issue after every step has been completed, there can be a lot that has to be undone in order to address it. This philosophy holds true with software, and it’s one that should be embraced. Bugs that are found and fixed early are less costly to address.

It’s more a practice of operational efficiency. Not only are we trying to get features implemented, but we are also focused on making sure that they’re implemented in a way that meets our overall quality goals and the needs of our users all along the way.

***Traditional disciplines in software development***

When we hear about quality engineers it’s not uncommon to hear about sub-categories that define the roles and responsibilities of an individual. We see, there are many different kinds of testing and strategies for ensuring that our product meets a certain quality bar. This is something we want to understand before we start looking for people to fill specific roles. The most common terms we will hear are black box, white box, and grey box testing.

Black box typically refers to testing an application through the user interface.

White box is used when testing is going to be done at the code level, where we validate everything that’s happening under the hood.

And finally, grey box is used when our testing is a mix of both. We are concerned about the application from the outside looking in, but we have some knowledge of the internals.

Let’s just assume that there are three core disciplines to a project. Product management is typically where our business goals are identified. Research is done to define target users, their needs, and what’s going to be built to address those needs. Development is where we take those business requirements, identify the best ways to fulfill them from an engineering perspective, and proceed with building our product. Quality assurance is where we are going to be working to ensure what we are building is meeting all of our goals.

When it comes to quality, it’s not just a focus on making sure that the product is functional. we are validating that the product is truly meeting our users needs and that from an engineering viewpoint, it’s being done in a way that we set out to do. Now, I’m a firm believer that ownership of the overall project is shared equally between these three disciplines. And that there’s going to be some overlap. This doesn’t mean that there are in inherent redundancies but rather parts of the overall project may be looked at by multiple people with different perspectives.

Since we’re focused on quality for this course, let’s look at the areas the quality discipline that overlap others. If we look here, we will see that there’s a portion of quality that overlaps product management. This is the area that we should be thinking about when we hear the term black box testing. Contrary to what some might lead we to believe, it’s not a function where purely manual testing is conducted to make sure that UI elements are functional. It’s more than that. This is where the quality discipline intersects with our business goals to ensure that they’re being met.

Black-box testing is one of the most commonly used terms when discussing quality assurance. The basic principle in this method of testing focuses on functionality without looking at the internals. It’s the process of looking at our product from the user interface and interacting with it as the user would. This type of testing, by definition, doesn’t have any specific concern about anything that’s happening at a lower level. The term comes from the concept that if we had a completely sealed black box, we wouldn’t be able to see what’s inside it. So we are only working with the exterior that’s available to us, for anything that we plan to test.

One common misconception is that black-box testers are only given manual tasks, where buttons are clicked to make sure that they produce a specific result. In reality, the term doesn’t imply that the role is purely manual in nature. A black-box tester can absolutely possess technical skills and perform automation tasks that require scripting knowledge. The term is used to indicate each test case we’ll be looking at the product from the outside. Regardless of that these specific responsibilities are, a good black-box tester will have an understanding of how all of the pieces of our application fit together.

**White box** as, i mentioned previously that it’s best to think about white box testing as being closely integrated with development. And that’s because it’s a very technical role. Sometimes referred to as a software developer in test, this individual is going to be focused on testing at the code level. And often, they’ll have other responsibilities that are related to testing, but may not test our application specifically. The term white box means that the container is completely transparent and we will be able to focus exclusively on the internals. A white box tester isn’t going to be concerned about the application UI or the overall usability at all.

Rather, they’re going to be working closely with developers to make sure that the tests they write validate all interactions and behaviors that ultimately result in functionality that’s presented to the user. So let’s say we are building a calculator. There’s going to be code that defines how the mathematical operations function. And our white box engineer is going to be focused on the expected values returned for every input. There’s going to be a lot of possible inputs and results. So a tester in this capacity is going to be tasked with systematically breaking down these possibilities in to finding a strategy to test that every component of our calculator is functioning as expected.

Its talk a little bit about grey box testing and what it is like our other box metaphors, we can visualize this by thinking about a tester approaching a box from a test perspective. Only in this case its semi-transparent which allows for a more in-depth understanding of the test cases that should be executed and what the results mean. I mentioned earlier that people often refer to grey box testing as a transition point for a black box tester where the focus starts to become more technical and is no longer purely focused on the application solely from the user interface.

That’s a key distinction. A lot of times when we hear about a quality manager seeking testers with grey box skills they’re looking to increase the technical knowledge on their team. They may have internal goals of fostering development of grey box skills among their black box testers and there’s a true advantage to this. We need to keep in mind that as methods for the delivery of a product make it easier to get in the hands of users, it’s not that hard to find somebody to use our product and provide feedback.

A black box tester would likely only be focused on test cases where they interact with the form and the feedback that they get from the UI. This is great if everything works as expected. But what if they enter in all the appropriate information into the form, click submit, and nothing happens? Well, a grey box tester might use something like a web debugger to monitor what the application is sending and then look for valid server response. They may see that when they click on the submit button the request is made from the client but there’s no response from the server.

This is important because it helps identify exactly where the problem is occurring which in this case would be the back end. They also may use a tool like this to modify the server response and verify how the client behaves in different situations. These are only a few small examples of what a grey box test case might look like. But the **important** takeaway is that at this level of testing, you’re extending your view to peek at the internals of your application which ultimately give you a better understanding where problems might occur and increase your ability to report these problems with greater accuracy.