Communicating Risk Because We Can't Test Everything

Jenny Bramble

WillowTree Apps, Inc
jenny.bramble@gmail.com
http://twitter.com/jennydoesthings

Abstract

The idea of testing everything is a popular one, in fact, many stakeholders think that's exactly what their quality teams do. It usually isn't and can't be; but how can teams communicate this? Join Jenny Bramble as she helps to pave the way using the language of risk-based testing. By defining risk in two simple parts, the team and project have a tangible and usable metric. She shares how to apply this metric and use it to determine where the team should focus testing, making it more effective and efficient whilst communicating that effort through the creation of a risk matrix.

As a result, risk becomes the right language for the team to communicate clearly and concisely with everyone involved in the project by using agreed-upon words and definitions.

Biography

Jenny Bramble ended up--as many of us do--falling in a quality assurance career after trying a few other routes. She came up through support and DevOps, cutting her teeth on that interesting role that acts as the 'translator' between customer requests from support and the development team. Her love of support and the human side of problems lets her find a sweet spot between empathy for the user and empathy for my team. She's done testing, support, or human interfacing for most of her career. She has been interested in risk her entire career, finding it the foundation that all QA is built upon.

1 Introduction

One of the hardest parts of my career in quality assurance has been telling someone--anyone--that I'm worried about a feature that they really want to release. I'm an emotional person and favor talking in pictures and feelings over logic and rationale. This would often leave me in the sorts of situations where everyone involved walked away upset.

"Hey," I'd say to one of the developers on my team. "I feel bad about this feature that we're trying to push out for the next release."

"What do you mean?" he'd ask.

"It's just...I have a bad feeling." I might wave my arms about or shrug.

He'd look at me, look at his computer, and decide that it wasn't worth his time to try and force me to make sense.

This scene would repeat over and over in my days as a junior QA person and I never really understood why they didn't seem to understand me. I thought I was communicating well, and after a while, people would trust my 'gut instincts' about our projects. Yet, every time we'd get a new developer, new product owner, new project manager, new quality assurance person, or new intern, the cycle would start again.

What was going on? It turns out I was trying to communicate risk to my stakeholders (in this case, my developers) and because we were using different subsets of language, I was not able to effectively communicate.

2 What is Risk?

Risk--or our feelings about risk--while essential to every aspect of a project is not easy to communicate. We use words that mean something to us, but that may mean something entirely different to another person. Considering this is a metric that we use to define our scope of testing, this is a gap in our communication that can be very detrimental to our projects. If we can't talk about risk in a concrete, meaningful way, we are losing one of the best tools in our QA toolbox.

So, then, what is risk? Can we give it a value that we can use to compare different features and use cases?

Yes!

Simply put, risk is the impact of failure and the likelihood that failure will occur. The intersection of impact and likelihood gives us that value.

Once we can determine the impact of the failure that may occur and the likelihood of that failure, we can create a scale that we will use to assign a value to the likelihood and the impact which can be combined to give us a value that we can assign as risk.

Let's start with a definition of "failure" and its impact.

3 Failure, Impact of Failure, and Probability of Failure

Failure is anything that has an impact on the business. The most obvious examples are bugs that cause catastrophe--users can't login or can't perform basic functions. But, there are a lot of layers to the impact that failure can have. When we've determined what our failure cases are, we can move on to talking about the impact of these failures.

The easiest types of impacts to see are the technical failures. Examples here are features not working or behaving in unexpected ways from something as simple as 'users can't log in' to a more complex scenario such as 'the data returned are subtly off from the expected value.' We see these failures every day as we navigate the web or use mobile applications on our phones. They're the most obvious of failures to end users.

But, there are other types of failures. You can have a failure that impacts the business side of the house: maybe there's a feature released too early. Or a rule that's not been properly implemented. Or worse--legal language missing from a site that brings your company out of compliance.

Most importantly and most often overlooked, we can also have a failure that impacts the morale of our users; an emotional failure if you will. This happens when users have their workflow interrupted or something that causes them to need to do extra work. Even a flow that improves the time they need to spend on a task can be disruptive if it's not communicated clearly.

When we finally can sit down and talk about all the ways a use case, feature, or application can fail, we can start to have a real meaningful discussion on what the impacts are. This is the first time in the process that we start to level set and align everyone's expectations together. Once this is started, you will find your teams have a much easier time communicating.

The next area to focus on is probability. Simply put: how likely is it that this use case, feature, or application will fail in the ways we've described?

There is not always an obvious answer to this question. As with the impact, we end up making an educated guess. This means that our first step is to educate ourselves.

Begin by asking questions. Has this feature failed before? How did it fail? When? Have we handled that use case?

Has something similar failed before?

Are there any outside influences acting on our team that may make this feature more likely to fail?

Do we have any root cause analysis documents or defects that can tell us about past failures? Have we asked any of our team members who've worked in this area of the code before about it?

How likely is this use case to be encountered by end users? How many end users may see the failure?

All of these answers will give us information that we can use to determine the likelihood of the item failing in our present day. There's always the chance that we'll be wrong, but our job is to make these educated guesses by collecting as much information as is feasible and translating those answers to the team in a reasonable way.

If you are looking at your item and are concerned that you're not getting the full picture, there's a few other things you can do as well.

First, remember that there are other types of risks. Perhaps you want to indicate that user familiarity is a factor that will increase the risk or likelihood that you will see issues. Or you are concerned about the number of merges a particular feature has and you'd like to use that to weight the risk value. If you have any industry specific risks that come into play, you can use those as well. Anything that gives you enough information to make a decision but not so much that you create a lot of chaos or swirl is worth noting.

Second, look for more or less granularity. If you are looking too closely at the details then you may not be getting a good picture of the true risks. Similarly, if you're going too broad, you may miss individual pieces that need more attention. Maybe there's a particular use case that has a high impact of failure yet the overall feature tends to have a lower impact as there's several ways to access it. This would be valuable to call out separately.

Once these things have been hashed out, we can start using our rating system. By defining what we mean when we say risk, we've already started to create a stronger team. Using the same words with the same meanings, we're able to accurately communicate to our team members.

4 Building a Risk Matrix

Begin by deciding how you'll represent your concerns or feelings on the impact and likelihood. A scale of 1 to 10 is most often used, though you can use anything that makes sense to your team. Values of high, low, medium or t-shirt sizing can also work. If you have a team that is particularly emotion driven, a set of emojis can also be a fantastic visual way to represent the impact, likelihood, and resulting risk.

Create a matrix with one row for each of your use cases or features and a column for each item that will contribute to the overall risk value and a column for the over-all risk.

Item/Use Case	**Impact of Failure**	**Probability of Failure**	**Risk Value**

As you can see, it looks pretty simple. This is by design. The more complex a risk matrix is, the less likely it is that it will be filled out in a timely, useful manner.

Enter in all the use cases or features you want to look at. Remember that for each risk assessment session, you should focus on a subset of features, since doing all of them at once will muddy the waters and you may find yourself rushing to just get numbers down.

Item/Use Case	**Impact of Failure**	**Probability of Failure**	**Risk Value**
Upload a song to your account			
Set a song to "fans only"			
Add lyrics for a song			

With the team or with yourself, discuss the rating system. What does a catastrophic failure look like for these scenarios? What does an acceptable failure look like? What about a failure that is bad, but not earth shattering?
Once you have those things defined, you can enter in values for the Impact of Failure column.

Item/Use Case	Impact of Failure	Probability of Failure	Risk Value
Upload a song to your account	10		
Set a song to “fans only”	6		
Add lyrics for a song	2		

In this scenario, uploading a song is our golden path--our most important workflow. If a user is unable to upload a song at all, that constitutes an all-hands-on-deck emergency. But if they can’t set a song to “fans only”, that’s not as big of an issue. It’s still bad and we need to resolve it, but it isn’t a severity one emergency.

If users are unable to add lyrics to a song, we are not as concerned. It’ll be fixed, but perhaps in the next build not as a hotfix.

Next, we look at the probability of failure. This is where most of the discussion happens. When I run a risk assessment session, I will often time box these discussions to 3-5 minutes. You can discuss it longer, but if you restrict the time (or timebox it), people will often start to agree on a number.

Item/Use Case	Impact of Failure	Probability of Failure	Risk Value
Upload a song to your account	10	4	
Set a song to “fans only”	6	8	
Add lyrics for a song	2	3	

Here, we’ve determined that uploading a song probably will not break, but it’s a workflow that we have in a lot of places and there’s a chance of it failing as a result of other work.

Setting a song to “fans only” is brand new functionality and we’ve never had any gatekeeping in place like this. Therefore, it has a high chance of breaking or being implemented incorrectly.

Adding lyrics has a low probability of failure. It’s a simple text page with no formatting and we haven’t touched that code path in a long time.

Now that we have this, we can multiply the values together and get a value for risk.

Item/Use Case	Impact of Failure	Probability of Failure	Risk Value
Upload a song to your account	10	4	40
Set a song to “fans only”	6	8	48
Add lyrics for a song	2	3	6

We can clearly see and now clearly communicate to our team where we should be focusing our testing efforts. Setting a song to “fans only” is the obvious choice, with uploading songs a close second.

Now that we have these numbers in our hands, we can take them to other parts of our cross functional teams and say ‘this is where our efforts will be focused and this is why.’

This lets us not only justify our decisions to our team but it can also let us argue for more resources such as more people working on a project or more time to complete the work we’ve been given. In short, risk drives these decisions and by clearly defining risk as impact of failure multiplied by probability of failure we can communicate risk via a risk matrix. This helps our team members make better, more informed decisions.

5 Summary

In summary, the language of risk-based testing, once it’s defined by your team, is a powerful tool to use for communication. You can supplement the definitions with a risk matrix and find that you and your team communicate more clearly, precisely, and will lead to better testing of your application. In turn, this leads to high quality and higher confidence overall.

6 References

Heuristic Risk-Based Testing by James Bach
http://www.satisfice.com/articles/hrbt.pdf

Risk-Based Testing: Test Only What Matters by Rajnish Mehta
https://www.stickyminds.com/article/risk-based-testing-test-only-what-matters-0

Risk Based Testing, Strategies for Prioritizing Tests against Deadlines by Hans Schaefer
http://www.methodsandtools.com/archive/archive.php?id=31

Wave of risk perception
http://www.a-sisyphean-task.com/2018/06/the-wave-of-risk-perception.html