We often see situations where we have applied the best ***testing techniques*** and processes, and yet the testing wasn't completed in time or with quality. It happens when we have not planned for risks in our testing process. In this tutorial, we will get a good understanding of Risks and it's various levels. We will also discuss how we can categorize risk in software testing basis their probability and impact.

What is Risk in Software Testing?

Risk is the possibility of an event in the future, which has negative consequences. We need to plan for these negative consequences in advance so we can either eliminate the risk or reduce the impacts.

From the Testing perspective, a QA manager needs to be aware of these risks so he/she can minimize the impact on the quality of the software. Does this mean that the QA manager should address every risk that the project could face? In an ideal world, YES, but in all practicality, he would never have the time and resources to plan for every risk. Therefore we need to prioritize risks that would have severe consequences on software. How do we do that? We do that by determining the level of risk.

Dimensions of Risk

There are two dimensions of ***Risks*** that we should know.

Probability *- Risk is always a possibility. The likelihood of risk is always between 0 % to 100 %. The probability can never be 0%; otherwise, risk will never occur. It can never be 100%; otherwise, it's not a risk; it is a certainty. E.g., We are hosting a website on a server that guarantees 99% uptime. What is the probability of the server going down? You guessed it right! It's 1 %.*
Impact *- Risk by its very nature has a negative impact. However, the size of the impact varies from one risk to another. We need to determine the impact on the project if the risk occurs. Continuing with the same example - What's the impact if the server goes down? Well, the site will not be accessible, so the impact is very high!*

Levels of Risk

Based on these two dimensions, we determine the level of risk.

Level of Risk in Software = Probability of Risk Occurring X Impact if risk occurred

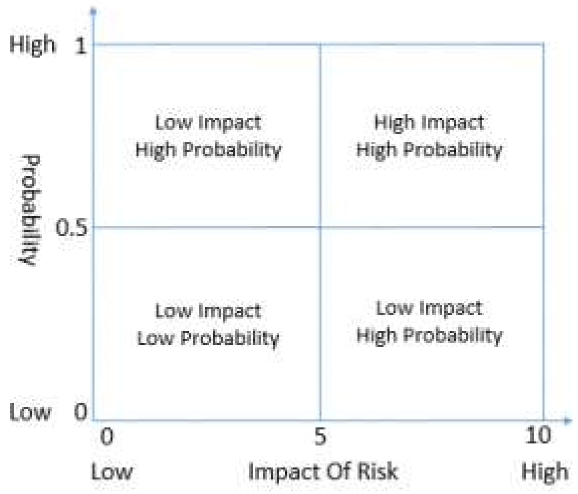
We can calculate the probability of risk between 0 - 1 with 0 depicting 0% occurrence and 1 depicting 100% occurrence. In this case, the classification of the impact is Low, Medium, and High. Some folks also classify it as Minimal, Minor, Moderate, Significant, and Severe. For the formula to calculate the level of risk, we can show the impact on a scale of 1 - 10 with 1 being the lowest impact and 10 being the highest impact. We can also use a range of 1-5, but irrespective of that, the core concept remains the same.

If we continue with our earlier example of server uptime of 99% - The Risk level will be calculated as :

Level of Risk = 0.1 X 10 = 1

0.1 is the probability of server going down (*1% will translate to 0.1*), and 10 is the impact on a scale of 1-10.

The Level of Risk calculation helps us in prioritizing risks. If we plot the probability and impact on a graph, we can classify the level of risk as below.



Let's understand these with examples - We will only discuss the concepts here. We will address the actual risk mitigation in our next article.

Low Impact Low Probability

These are the risks that have a low probability of occurrence, and the impact on software is pretty low.

E.g., Consider an e-commerce website that provides a Chat option so customers can chat with the service desk executive if they face any issues. Chat integration is a third-party plug-in, and it gives an uptime guarantee of 99%.

> *What's the probability of chat service not available? It's 1% which is pretty low*
> *What's the impact if chat service goes down? The effect on software is pretty low, provided only a few users use chat service. Or there are other options to reach customer service executives (E.g., phone, and emails).*

Such risks, which are low probability and the low impact, can be ignored. There isn't much value, add spending time on these risks.

Low Impact High Probability

These are the risks that have a high probability of occurrence, but the impact on software is pretty low.

E.g., We are migrating users from one website to another. The phone number format of both sites is different. As such, the probability of users losing their phone numbers in their profile is pretty high. However, As the phone number is not a mandatory field, it will not impact any user journeys. Also, a user can go ahead and update the phone number in the new format in My Account. Hence, the impact on software is low.

Such risks don't need much mitigation planning. These, however, need to be monitored to ensure that the impact remains low (*What if the phone number becomes mandatory and user journey gets blocked ?*)

High Impact Low Probability

These are the risks that have a low probability of occurrence, but the impact on software is pretty high.

Consider that we are hosting our test website on a server that guarantees 99.9 % uptime.

> *What's the probability of the server going down? It's 0.1 %, which is pretty low.*
> *What's the impact if the server goes down? The website will not be accessible, and testing will completely stop. Do you see that it's a very high impact?*

For such situations, we need to ensure that we have a mitigation plan if the risk does occur. It could be executing the tests in a different environment for the time the original server is down.

High Impact High Probability

These are the risks that have a high probability of occurrence, and the impact on software is pretty high as well.

Consider a situation where we are planning for testing software, and the timelines are very aggressive. The testing requires 10 Appium skill set resources; however, the availability of this skill set is very minimal in the organization.

> *What's the probability that we will not get the required resources on time? Well, It's pretty high, given that this skill set is scarce, and deployment of existing resources in on-going projects has finished already.*
> *What's the impact if we don't get these resources in time? As the timelines are pretty aggressive, the impact on test completion will be pretty high!*

High Impact and High Probability is the highest level of risk in software testing, and maximum planning and attention should go to this bucket. These risks have serious potential to derail testing thoroughly, and it could lead to delays in test completion or poor software quality.

For our current example, one mitigation could be to hire this skill set from the market. We can also hire contractors for a short duration to help in execution. As you would realize, the earlier we identify these risks, the easier it is to put the mitigation plan in place.