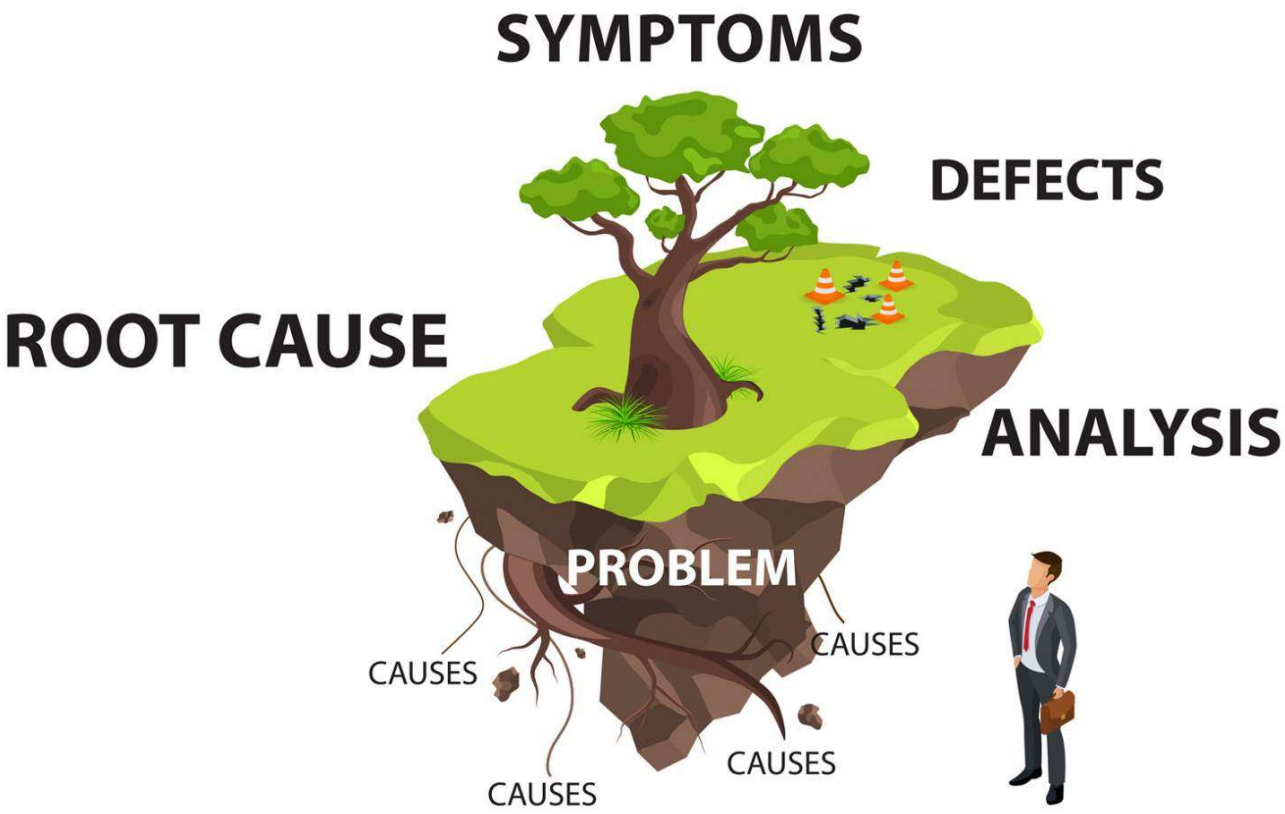
We all have read that "***An ounce of prevention is worth a pound of cure***". If you translate these words in the software context, it means that it gets more expensive to fix the defects if they linger on for long in a process. So ***Root Cause Analysis*** is required to find the causes of problems to provide appropriate solutions.

We will focus on the following points:

What is Root Cause Analysis?

Root Cause Analysis is a systematic approach to identifying the underlying causes of an incident. It helps in taking appropriate steps, so the problem is addressed. In an ideal world, the software should have Zero defects, and it should not show any failure. Practically, it's never possible. By effective QA processes, we can ensure that software has minimum defects. We can also ensure that there are the right techniques that can be followed to find the underlying cause of any defect. This process of identifying WHY the problem has occurred in the software is called ***Root Cause Analysis*** (*RCA*).

The easiest way to understand root cause analysis is to think about everyday problems. If we are sick, we will go to a doctor and ask them to look for the cause of our illness. If our car stops working, we will ask a mechanic to find the root cause of the problem. To solve or analyze a problem, we will need to perform a root cause analysis and find out precisely what the reason is and how to solve it.



The root cause analysis is like a chain of events that go backward. It starts from the last possible action to the previous one, so on and so forth. It continues until it reaches the beginning of the problem. Hence, it's called reverse engineering.

There are a few essential questions to ask: ***WHAT, WHY, WHEN, HOW.***With the help of these questions, we can delve into each phase of the software life cycle. We do it to accurately track the origin of the defect and the point of time when it injected into the system.

Why Defect Occurs?

Before we dive further into Root Cause Analysis, let's first understand the common factors responsible for defects in software:

> ***Inadequate requirements***: *The success of any software application depends on the understanding of development teams on client requirements. Unclear requirements and misunderstanding of the requirements are two main factors that cause software defects.*
> ***Programming errors***: *Programmers, like any other person, can make programming mistakes. Inexperienced programmers or programmers without proper domain knowledge can introduce*

simple errors during coding. The lack of simple coding practices, unit tests, debugging are some of the common reasons why most problems appear at the stage of development.
Communication gap: *Defects are introduced in the development stage if the exact requirements are not communicated correctly to the development teams. Sometimes miscommunication between the developers and testers may also lead to defects in software.*
Time Pressures: *Scheduling of software projects is not an easy job. When the deadline approaches and the crisis arrives, mistakes happen. If there is not enough time for proper design, coding, and testing, it is quite evident that defects will be introduced and missed in testing*
Inappropriate environment: *Inappropriate environment such as hardware or software incompatibility leads to defects in the software. Often test data present in one environment is different than another environment. Therefore, some of the defects go undetected.*

How to do Root Cause Analysis?

Root Cause Analysis is a 4 step process. We need to ask 4 Questions that will summarize the ***RCA process***:

WHAT? *The First step is to identify WHAT is the problem. If we are not clear on the problem statement, we will never be able to find the root cause. E.g. The customers have reported that they are not able to place any orders.*
WHEN? *The next step is to find out WHEN the problem has happened. For our current example, when the customers were trying to place an order, they got an error message that "**Orders cannot be processed**". It happened between 3.30 pm to 4 pm, and an error message comes when customers click on the place order button.*
WHY? *The next step is to identify WHY the problem occurred. Based on the information identified in WHAT and WHEN, a detailed analysis takes place to identify the underlying root cause. For our current example, the underlying cause was that the payment systems were down between 3.30 pm to 4 pm. Which, in turn, interfered in placing the orders.*
HOW? *The last step for Root cause analysis is to find out HOW we can ensure that the problem does not occur again. For our current example, we can ensure that there is an alert mechanism that can send emails when any of the systems are down. Displaying a message on the website that there are some issues with order placement, and the team is working to fix it, can help the users. It will ensure that there is no impact on the User experience, and the technical team gets timely alerts to correct the problem.*

Defects can affect a software product or its functionality, such as the failure of a feature/functionality or the complete system failure. It is resulting in the loss of money, time, and reputation. So Root Cause Analysis discovers what went wrong. Usually, we use RCA as a way to diagnose problems. However, it can be equally effective in finding the root cause of success. This type of analysis can help prioritize and proactively fix future errors that occur due to the same root cause.

Examples of Root Cause Analysis

We have seen how root cause analysis helps in identifying the problem and take the next steps. It's also essential that the industries utilize these RCAs to improve their chances of success.

*Let us consider the case of Disney's Lion King. The Disney company launched its first multimedia CD-ROM game for children, "**The Lion King Animated Storybook**", in the fall of 1994. Although many other companies had been marketing programs for children for years, this was Disney's first adventure in the market and got massive promotions and publicity. The sales were huge. What happened, however, was a great debacle.*

On December 26, the day after Christmas, Disney customer service phones began ringing, ringing, and ringing. Soon, telephone support technicians got a large number of calls from angry parents with crying children who couldn't make the software work. The problem was the defected software that would not work on a top-rated PC platform. It turns out that Disney could not test the software in a broad representation of the different PC models available in the market. The software worked on a few systems that Disney programmers probably used to create the game, but not on the most common systems that the general public had.

However, nobody reported the defect even after the creation & selling of thousands of CD-ROMs. Disney ended up paying by product returns and replacement CD-ROM. They also incurred other costs associated with debugging, repair, and test cycle. This defect affected the company's reputation and financial status.

The company found the root cause of the problem was that the software didn't work on a top-rated PC platform. Following the right test approach and testing the software on the top 5 platforms that the consumers used, could have made the detection of this issue easier before the mass printing of the CDs. The other approach was the testing of the software Beta (*where a group of customers checks it*). Which, in turn, could ensure the detection of the issue before the launch.

The majority of industries follow this simple example of RCA since then. All gaming companies have ensured that they test their game on top 5 Operating Systems available in the market.

I hope you got a great understanding of Root Cause Analysis and it's importance in the success of current and future projects.