RELIABILITY CONCEPT AND OVERVIEW

Arrelic Insights

www.Arrelic.com

Overview

In recent years, manufacturing companies, government agencies, and civilian populations have placed a greater emphasis on reliability. With recent concerns about government spending, departments are attempting to buy systems that are more dependable and need less maintenance. We, as customers, are primarily interested in purchasing goods that last longer and are less expensive to maintain, , i.e., have a higher reliability. The benefits of high product, variable, or device reliability are self-evident:

- Improved customer loyalty
- Increased revenue
- Increased protection
- Lower warranty costs
- Lower maintenance costs, etc.

Basic Concepts of Reliability

The term "reliability" refers to a broad definition. When we expect anything to act in a certain way, we use it. One of the measures used to assess quality is reliability. It's a user-oriented consistency aspect that has to do with how the system works. Intuitively, if a system's users rarely encounter failure, the system is thought to be more reliable than one that fails often.

A fault-free system is regarded as extremely reliable. Building a reliable system is a challenging task. If the frequency of failure is "acceptable," then a flawed system may be considered reliable. Few concepts of reliability are Fault, Failure, Error, Time.

Failure

When the observed outcome of a program execution differs from the intended outcome, it is considered to be a loss. It's a complex situation. For a failure to occur, the software must be running. The word "failure" refers to the program's actions. It may involve items like a lack of performance attributes and a long response time. For, example, a faulty code block may be the source of a failure.

Fault

A fault is the determined cause of failure. A flaw is a flaw in a program that causes it to crash when it is run under such conditions. Failures can be caused by a variety of conditions, or the conditions can be replicated. As a result, a single fault can cause multiple failures. A fault is not a function of the program's execution or actions, but rather of the program itself. It's what we're actually talking about when we say "worm" in general. When a programmer makes a mistake, a fault is made.

Software reliability focuses solely on faults that have the potential to cause failures, identifying and eliminating faults that cause failures, and implementing fault tolerance strategies to prevent faults from causing failures or to mitigate the impact of failures that do occur.

Error

It is characterized as an error or omission in human action that results in a fault in a system or part. Misinterpretation of user specifications in a product specification, inaccurate translation, or lack of a requirement in the design specification.

Time

In the formulation of reliability, time is a central concept. We call a device less reliable if the time between two consecutive failures is short. There are two types of time that are considered.

a) Execution time (t): The execution time of a software system is the amount of time a processor spends executing the software system's instructions. Since a processor can often execute code from multiple programs, their individual execution times must be regarded separately.

b) Calendar time (t): Calendar time is the time that people are used to in terms of years, months, weeks, days, and so on. Calendar time is useful for expressing reliability because it allows managers and software developers to see the date when the system achieves its reliability objectives.

In other words, you are calculating calendar time if you start a stopwatch when you start a program and check it when the program is over. This is the same as "actual" time. If the computer has a lot of stuff running at the same time, the execution time would obviously be much shorter than the calendar time.

RAM – Reliability, Availability, and Maintainability

The term "reliability" is sometimes used to refer to a broad definition that encompasses both availability and maintainability. In its most basic form, reliability is concerned with the likelihood of a failure occurring within a given time frame, while availability is a measure of something being in a state (mission capable) ready to be tasked (i.e., available).

Maintainability is a parameter that considers how the current system can be restored after a failure, as well as concepts such as preventive maintenance and Built-In-Test (BIT), required maintainer skill level, and support equipment. The maintainability requirement must be invoked when dealing with the availability requirement since some form of repair and restoration to a mission-capable state must be included.

Clearly, logistics and logistic support methods are intertwined and play a role in the availability criterion as dependent variables. Sparing techniques, maintainer instruction, maintenance manuals, and the identification of appropriate support equipment are all examples of this. The relationship between RAM requirements and logistics support dependencies demonstrates how RAM requirements have a direct effect on sustainment and overall LCC. RAM specifications, in simple terms, are the higher-level, overarching requirements that are defined at the overall system level. Decomposing these higher-level criteria into lower-level design-related quantitative requirements including Mean Time Between Failure/Critical Failure (MTBF or MTBCF) and Mean Time To Repair is often required (MTTR).