We use the software in day-to-day life, such as at home, work, banking, shopping, etc. However, at times, the software does not work as expected. For instance, website not loading correctly, an error on a bill, a delay in credit card processing, are typical examples of problems that may happen because of errors, defects, and failures in the software. Today we will discuss the common terminologies that we use when software doesn't work as expected, i.e., ***Error, Defect, and Failure***.

Introduction to Error, Defect, and Failure

Let's try to understand the inter-relation between ***Error, Defect***, and ***Failure***:

It is well said by Thomas Muller "***A person can make an error (mistake), which produces a defect (fault, bug) in the code, in software or a system, or a document. If the execution of the defect in code happens, the system will fail to do what it should do (or something it shouldn't), which causes a failure***".

Let's take an example of an organization that developed a new application named "***Saff Promotion System***" for their annual appraisal. But employee satisfaction even after the appraisal was low. Because they considered it not up to the mark. The management, then, decided to analyze the root cause of this dissatisfaction.

Therefore, they backtracked the process and found that the software marked full day leave for the employees who reached office after 10 a.m. This was due to some coding errors. The tester also skipped these errors in coding. So, the software with this defect went to production. Which, in turn, caused a general degradation and failure of the system.

The figure below shows the interrelation between Error, Defect, and Failure.

Errors:

The ***Error*** is a human mistake. An Error appears not only due to the logical mistake in the code made by the developer. Anyone in the team can make mistakes during the different phases of software development. For instance,

BA (business analyst) may misinterpret or misunderstand requirements.
The customer may provide insufficient or incorrect information.
The architect may cause a flaw in software design.
People on the team can also make mistakes due to unclear or insufficient requirements, time pressure, lethargy, or other reasons.

Let us observe the basic types of errors in software testing:

Types of Error

User Interface Error*: These are the errors that generally appear during user interaction with the system. Such as missing or incorrect functionality of the system, no backup function or reverse function available, etc.*
Error handling error*: Any error that occurs while the user is interacting with the software needs precise and meaningful handling. If not, it confuses. Therefore, such errors are known as* ***error handling errors****.*
Syntactic error*: Misspelled words or grammatically incorrect sentences are Syntactic errors and are very evident when testing the software GUI.*
Calculation errors*: These errors occur due to bad logic, incorrect formulas, mismatched data type, etc.*
Flow control error*: Errors concerning passing the control of the program in an incorrect direction where the software program behaves unexpectedly are flow control errors. Such as the presence of an infinite loop, reporting syntax error during run-time, etc.*
Testing errors*: It implies the errors that occurred when implementing and executing the test process. For example, bug scanning failure, inefficiency in reporting an error or defect.*
Hardware errors*: Such errors are related to the hardware device. Such as no availability and no compatibility with the device.*

Defect:

A ***Defect*** is a variance between expected and actual results. An Error that the tester finds is known as Defect. A Defect in a software product reflects its inability or inefficiency to comply with the specified requirements and criteria and, subsequently, prevent the software application from performing the desired and expected work. The defect is also known as ***Fault***.

Types of defects:

Severity Basis:

Severity defines the degree of impact. Therefore, the severity of the defect reflects the degree or intensity of a particular defect to adversely impact a software product or its operation. Based on the severity metric, a defect falls under the following categories:

. ***Critical****: Defects that are "**critical**" require immediate attention and treatment. A critical defect directly affects the essential functionalities which can otherwise affect a software product or its large-scale functionality. For instance, failure of a feature/functionality or collapse of the entire system, etc.*
. ***Major****: Defects, which are responsible for affecting the main functions of a software product are Major Defects. Although, these defects do not result in the complete failure of a system but may bring several primary functions of the software to rest.*
. ***Minor****: These defects produce less impact and have no significant influence on a software product. The results of these defects are visible in the operation of the product. However, it does not prevent users from executing the task. The task can be carried out using some other alternative.*
. ***Trivial****: These types of defects have no impact on the operation of a product. Hence, sometimes, we ignore and omit them. For example, spelling or grammatical errors.*

Probability Basis:

One more angle to see a defect in a software application is the probability that it will occur, and chances that the user will find it. Depending on the likelihood or the possibility of a defect in a software product in terms of percentage is classified in the following ways:

. ***High****: Almost all users of the application can track the presence of defects. This indicates a high probability.*
. ***Medium****: Half of the users can trace the presence of defects in a software product.*
. ***Low****: In general, no user detects it, or only a few users will be able to detect it.*

Priority Basis:

Defects also have a business perspective comparison. The rectification of some defects must happen first. Likewise, some can solve at a later stage. Just like a business where everything happens according to the current need and demand of the market. Just like the probability base, priority classification also occurs in the following ways:

***High**: The high priority defines the most critical need of a company to correct a defect. This should happen as soon as possible, in the same compilation*.
***Medium**: Medium priority defects are next to high priority. And any next version or release of a product includes addressing them*.
***Low**: This type of defect does not need to be corrected individually. Consideration of repairing these types of defects, along with any other defects is voluntary*.

Failure:

Failure is a consequence of a Defect. It is the observable incorrect behavior of the system. Failure occurs when the software fails to perform in the real environment.

In other words, after the creation & execution of software code, if the system does not perform as expected, due to the occurrence of any defect; then it is termed as Failure. Not all Defects result in Failures; some remain inactive in the code, and we may never notice them. Failures also occur due to the following reasons:

Any physical damage or overheating in the hardware can cause the whole system to fail.
If the software is not compatible with the hardware, then also the system performs unexpectedly.
Failures also happen by environmental conditions like a radiation burst, a strong magnetic field, electronic fields, or pollution could cause faults in hardware or software.

Not all unexpected test results are failures:

Failures can also

Happen due to a human error in interacting with the software, like entring an incorrect input value, or misinterpreting an output.
Occur when someone deliberately tries to produce system failure or cause malicious damage.
Happen because of the mishandling of test data, test environment, etc. Such conditions are known as defects, but they aren't actually a Defect.
Sometimes, tests that result in undetected defects can also cause failure.

What are the causes of Errors in Software?

Here we will discuss some possible causes of these errors.

***Time pressure**: At times, software development happens under limited / insufficient resources with unrealistic deadlines. Developers do not have enough time to test their code before delivering it to the testing team. Which, in turn, introduces errors.*
***Human fallibility**: Human beings are prone to make mistakes. It would be foolish to expect the software to be perfect, and without any flaws in it! Ironically, we have not yet discovered any other non-human agent that can develop software better than humans. Therefore, we continue to rely on human intelligence to develop software. Thereby, increasing the possibility of errors in it.*
***Inexperienced or insufficiently skilled project participants**: Allocation of correct work to the correct resource is fundamental for the success of any project. Team members should be assigned a task according to their skills and abilities. An inexperienced project participant may*

make mistakes if they don't have proper knowledge of the work. For example, a resource having a good understanding of the database but having limited knowledge of HTML/CSS is not suitable for designing a website.

Miscommunication between project participants*: Improper coordination & poor communication between various departments in a project can result in disrupted progress. Conflicts can arise each time a project participant misinterprets or misunderstands the words or actions of another. For example, the business analyst does the requirement gathering, and then some other team member does the documentation of the requirements. Any miscommunication between the two can lead to incomplete/wrong requirement document, which, in turn, affects the design of the project*.

The complexity of the code, design, architecture, or the technology to be used*: As the complexity of the program, concerning code, design or technology increases, the software becomes more critical and more bugs appear. It is because our brains can only deal with a reasonable amount of complexity or change. Our minds may not be able to process complex information like the form of design, architecture or technology, etc. Therefore, resulting in low quality and erroneous coding*.

Misunderstandings about intra-system and inter-system interfaces*: There are high chances of error while establishing an* ***intra-system****, and* ***inter-system*** *interfaces. Let's try to understand what is* ***intra-system*** *and* ***inter-system*** *interfaces mean:-*

- ***Intra-system*** *interface- It implies the integration of different modules/features within a system/application. For example, in an online shopping portal, we have three modules:* ***online order, shipping, and supply chain****. These three modules form the intra-system for the online shopping portal. When these different modules are combined, there is a possibility of errors in the final product.*
- ***The inter-system*** *interface- It is the compatibility of an application with other applications when operated together. For example, compatibility between smartphones and tablets while data transfer via Bluetooth.*

New, unfamiliar technologies*: Sometimes, the developers and testers need to develop an application using a technique that is unknown to them. Lack of proper knowledge and understanding can lead to errors. At that time, they require a certain level of R & D, brainstorming, and training to reach a reliable solution*.

Environmental conditions*: Natural disasters, such as outbreaks of fires, floods, lightning, earthquakes, etc. can affect the computers. For example, a system may not work correctly if the software inside is affected by radiation, electromagnetic, or pollution.*

After going through this article, I am sure you will agree with me that various problems and discrepancies encountered during the software process are interdependent and interconnected. Generally, the appearance of one leads to the introduction of another. Which, in turn, affects the functionality of the software and thus, leads to unexpected results.