**DEFECTS AND DEFECT LIFE CYCLE/BUG LIFE CYCLE**

## What is Defect/Bug?

A **defect** is an **error** or a **bug**, in the application which is created. A programmer while designing and building the software can make **mistakes** or **errors**. These mistakes or errors mean that there are ***flaws*** in the software. These are called **defects**.

## What is Defect Life Cycle?

***Defect life cycle***, also known as***Bug Life cycle*** is the journey of a defect cycle, which a defect goes through during its lifetime. It varies from organization to organization and also from project to project as it is governed by the software testing process and also depends upon the tools used.

Diagram

Description automatically generated

### Defect Life Cycle includes the following stages:

***New:*** When a defect is logged and posted for the first time. Its state is given as new.

***Assigned***: Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team. There can be two scenarios, first that the defect can directly assign to the developer, who owns the functionality of the defect. Second, it can also be assigned to the Dev Lead and once it is approved with the Dev Lead, he or she can further move the defect to the developer.

***Open:*** Its state when the developer starts analyzing and working on the defect fix.

***Fixed***: When developer makes necessary code changes and verifies the changes then he/she can make bug status as ‘Fixed’. This is also an indication to the Dev Lead that the defects on Fixed status are the defect which will be available to tester to test in the coming build.

***Retest:*** At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.

Once the latest build is pushed to the environment, Dev lead move all the Fixed defects to Retest. It is an indication to the testing team that the defects are ready to test.

***Reopened:***  If the bug still exists even after the bug is fixed by the developer, the tester changes the status to “reopened”. The bug goes through the life cycle once again.

***Deferred:*** The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are [*priority*](https://istqbexamcertification.com/what-is-the-difference-between-severity-and-priority/)of the bug may be low, lack of time for the release or the bug may not have major effect on the software.

***Rejected:*** If the developer feels that the bug is not genuine, developer rejects the bug. Then the state of the bug is changed to “rejected”.

***Duplicate :*** If the bug is repeated twice or the two bugs mention the same concept of the bug, then the recent/latest bug status is changed to “duplicate***“.***

***Closed:***Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, tester changes the status of the bug to “closed”. This state means that the bug is fixed, tested and approved.

***Not a bug/Enhancement:***  The state given as “Not a bug/Enhancement” if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of color of some text then it is not a bug but just some change in the looks of the  application.

## Tips to write a good bug report!

One of the most important skills that you need to have in your tester’s tool box is the ability to write a good bug reports. Finding defects is only part of the job, because if developers can’t reproduce the bugs you find, they’re going to have a very tough time fixing them. Your bug tracking software should include a number of mandatory fields to ensure that testers give a complete account of the defect they encountered. And testers should hone their descriptive skills.

### Good bug reports contain:

#### *****Descriptive Title*****

Everything starts with a title. It must be clear and descriptive, so that you get an idea of what it is about at a glance.

#### *****Concise Description*****

Try to remain clear and concise in your description. Describing something accurately and concisely is a skill that develops with practice. Remember that the person reading your description has not seen the bug and try not to make assumptions.

#### *****Actual Result*****

You have to make it clear what Actual Result you are getting while performing the Test Execution.

#### *****Expected Result*****

You have to make it clear what Expected Result you should get as per the requirements while performing the Test Execution.

#### *****Details about the project and version*****

It’s not unusual for testers to work on multiple projects at the same time, and sometimes they’ll share a database, so ensuring that your bug is listed in the correct project, and section within that project, is vital. You also need to get the software version right. Sometimes a bug will be fixed when a different defect is addressed, or simply by some change in the newest version of the software. If the version is wrong, developers may be embarking on a wild goose chase.

#### *****Platform Details*****

Any background information you can provide about your environment will help developers track that bug down. What device were you using? What operating system was it running? What model was it? What browser did you use? Every detail you can give about the platform will help.

For Example : Device Type, Model,Browser Version and OS Version.

#### *****Defect Type and Severity*****

These fields go hand-in-hand. A functional bug will generally be treated more seriously than a suggestion. Developers also need to know how severe the bug is. No product ships with zero defects, so having bugs categorized correctly in terms of type and severity really helps the decision-making process with regards to what gets fixed and what doesn’t. So it is always better to mention the Priority and Severity of the Bugs.

#### *****Steps to Reproduce (this is very important!)*****

You want to give a step-by-step account of exactly what you did to find any defect. Sometimes you can use software tools that catch your key strokes, or record screenshots or video files as you test, other times you will be writing from memory, so take notes as you go. Make sure that you test your own steps again before submitting the bug.

Give an indication of reproducibility as well. Sometimes you’ll be confident about your steps, but upon repeating them the bug won’t necessarily occur. If it happens every time, at random, or only once, then the developer needs to know that.

#### *****Visual Attachment*****

Supporting material is always gratefully received by those tasked with fixing defects. Usually this will be screenshots, but it can include audio and video files. Keep in mind this is often what developers look at first, before reading the text you provided. If you can convey the issue in a single screenshot, you’re helping save precious time!

#### *****Tags & Links*****

It can be a good idea to use descriptive tags that enable you to filter the database and find groups of related bugs. Sometimes you’ll want to include another bug ID or a link within your defect report to something that you feel is strongly related or similar, but not similar enough to be a duplicate.

#### Assignee

It’s usually going to be up to the lead developer or management to assign bugs, but on occasion you might receive instructions about who to assign to. Leaving unassigned or wrongly assigned bugs in the database is a dangerous thing because they can potentially slip through the cracks. Make sure you’re clear on the expectations for assigning.

## Bug Severity

**Bug Severity** or Defect Severity in testing is a degree of impact a bug or a Defect has on the software application under test. A higher effect of bug/defect on system functionality will lead to a higher severity level. A Quality Assurance engineer usually determines the severity level of a bug/defect.

## What is Priority?

**Priority** is defined as the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.  
Business/Stakeholders directs/guides in which priority the issues should be resolved.

## Types of Severity

In Software Testing, ***Types of Severity*** of bug/defect can be categorized into four parts:

* **Critical**: This defect indicates complete shut-down of the process, nothing can proceed further.
* **Major**: It is a highly severe defect and collapses the system. However, certain parts of the system remain functional.
* **Medium**: It causes some undesirable behavior, but the system is still functional
* **Low**: It won't cause any major break-down of the system.

## Priority Types

***Types of Priority*** of bug/defect can be categorized into three parts :

* **Low:**The Defect is an irritant but repair can be done once the more serious Defect has been fixed
* **Medium:**During the normal course of the development activities defect should be resolved. It can wait until a new version is created.
* **High:**The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed.

## Priority vs Severity: Key Difference

| **Priority** | **Severity** |
| --- | --- |
| * Defect Priority has defined the order in which the developer should resolve a defect | * Defect Severity is defined as the degree of impact that a defect has on the operation of the product |
| * Priority is categorized into three types   + Low   + Medium   + High | * Severity is categorized into five types   + Critical   + Major   + Minor   + Cosmetic |
| * Priority indicates how soon the bug should be fixed | * Severity indicates the seriousness of the defect on the product functionality |
| * Priority of defects is decided in consultation with the manager/client | * QA engineer determines the severity level of the defect |
| * Priority is driven by business value | * Severity is driven by functionality |
| * Its value is subjective and can change over a period of time depending on the change in the project situation | * Its value is objective and less likely to change |
| * High priority and low severity status indicates, defect have to be fixed on immediate bases but does not affect the application | * High severity and low priority status indicates defect have to be fixed but not on immediate bases |
| * Priority status is based on customer requirements | * Severity status is based on the technical aspect of the product |

**POSSIBLE COMBINATION OF PRIORITY AND SEVERITY EXAMPLE**

**HP HS- User is not able to login into the Application  
HP LS- Name/Logo of the product is not correct.  
LP HS- Home page link which is less accessed is not working/404 is coming  
LP LP- Any textual issues in the Application**