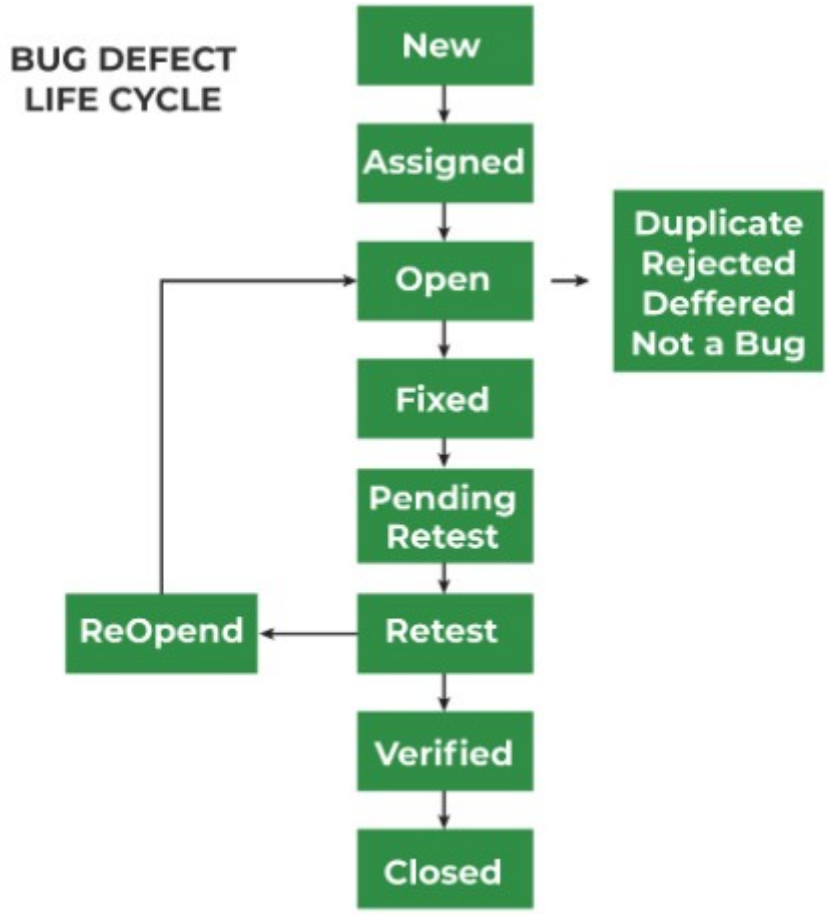
Defect Life Cycle:

Objectives:

The journey of the Defect Cycle varies from organization to organization and also from project to project because development procedures and platforms as well as testing methods and testing tools, differ depending upon organizations and projects.

The number of states that a defect goes through also varies depending on the different tools used and processes followed during the software testing.

The objective of the defect lifecycle is to easily coordinate and communicate the current status of the defect and thus help to make the defect-fixing process efficient.



1. New

When any new defect is identified by the tester, it falls in the 'New' state. It is the first state of the Bug Life Cycle. The tester provides a proper Defect document to the Development team so that the development team can refer to Defect Document and can fix the bug accordingly.

2. Assigned

Defects that are in the status of 'New' will be approved and that newly identified defect is assigned to the development team for working on the defect and to resolve that. When the defect is assigned to the developer team the status of the bug changes to the 'Assigned' state.

3. Open

In this 'Open' state the defect is being addressed by the developer team and the developer team works on the defect for fixing the bug. Based on some specific reason if the developer team feels that the defect is not appropriate then it is transferred to either the 'Rejected' or 'Deferred' state.

4. Fixed

After necessary changes of codes or after fixing identified bug developer team marks the state as 'Fixed'.

5. Pending Retest

During the fixing of the defect is completed, the developer team passes the new code to the testing team for retesting. And the code/application is pending for retesting on the Tester side so the status is assigned as 'Pending Retest'.

6. Retest

At this stage, the tester starts work of retesting the defect to check whether the defect is fixed by the developer or not, and the status is marked as 'Retesting'.

7. Reopen

After 'Retesting' if the tester team found that the bug continues like previously even after the developer team has fixed the bug, then the status of the bug is again changed to 'Reopened'. Once again bug goes to the 'Open' state and goes through the life cycle again. This means it goes for Re-fixing by the developer team.

8. Verified

The tester re-tests the bug after it got fixed by the developer team and if the tester does not find any kind of defect/bug then the bug is fixed and the status assigned is 'Verified'.

9. Closed

It is the final state of the Defect Cycle, after fixing the defect by the developer team when testing found that the bug has been resolved and it does not persist then they mark the defect as a 'Closed' state.

Example of Bug Life Cycle:

Step 1: New

While testing the search functionality on the e-commerce website, you, as the tester, notice that the search results page is blank when trying to search for a product. You document this issue by creating a defect report that includes details of the problem and the steps to reproduce it. Once the defect is reported, it enters the New state, indicating that the issue has been logged and is awaiting review.

Step 2: Assigned

After the defect is logged, the development team reviews it. Upon assessing the issue, they assign the defect to a developer for further investigation. At this point, the defect status changes to Assigned, meaning it is now the developer's responsibility to address the problem.

Step 3: Open

The developer begins investigating the issue and identifies that the root cause is a bug in the backend code, specifically related to how search queries are processed. The status of the defect is updated to Open, indicating that the developer is actively working to fix the problem

Step 4: Fixed

Once the developer resolves the issue by correcting the faulty backend code, the defect is marked as Fixed. The developer has made the necessary changes, and the fix is now ready for testing.

Step 5: Pending Retest

After fixing the bug, the developer submits the updated code for retesting. The status changes to Pending Retest, as the code now needs to be tested to ensure that the defect has been successfully addressed and that the functionality works as expected.

Step 6: Retest

As the tester, receive the updated code and retest the search functionality. After running the test, you find that the search results page now displays the correct products, confirming that the issue has been fixed. The defect's status is updated to Retest, indicating that the tester is validating the fix.

Step 7: Verified

Since the issue is no longer present after retesting and the functionality works as expected, you confirm that the bug is indeed resolved. The defect is then marked as Verified, showing that the tester has validated the fix and the bug is no longer an issue.

Step 8: Closed

After confirming that the defect is fully resolved and no further issues are present, you close the defect. The status is updated to Closed, indicating that the bug is resolved, and the life cycle is complete.

Another Defect States available in Defect Life Cycle:

1. Rejected: If the developer team rejects a defect if they feel that defect is not considered a genuine defect, and then they mark the status as 'Rejected'. The cause of rejection may be any of these three i.e Duplicate Defect, NOT a Defect, Non-Reproducible.

2. Deferred: All defects have a bad impact on developed software and also they have a level based on their impact on software. If the developer team feels that the defect that is identified is not a prime priority and it can get fixed in further updates or releases then the developer team can mark the status as 'Deferred'. This means from the current defect life cycle it will be terminated.

3. Duplicate: Sometimes it may happen that the defect is repeated twice or the defect is the same as any other defect then it is marked as a 'Duplicate' state and then the defect is 'Rejected'.

4. Not a Defect: If the defect has no impact or effect on other functions of the software then it is marked as 'NOT A DEFECT' state and 'Rejected'.

5. Non-Reproducible: If the defect is not reproduced due to platform mismatch, data mismatch, build mismatch, or any other reason then the developer marks the defect as in a 'Non-Reproducible' state.

6. Can't be Fixed: If the developer team fails to fix the defect due to Technology support, the Cost of fixing a bug is more, lack of required skill, or due to any other reasons then the developer team marks the defect as in 'Can't be fixed' state.

7. Need more information: This state is very close to the 'Non-reproducible' state. But it is different from that. When the developer team fails to reproduce the defect due to the steps/document provided by the tester being insufficient or the Defect Document is not so clear to reproduce the defect then the developer team can change the status to "Need more information'. When the Tester team provides a good defect document the developer team proceeds to fix the bug.

Severity and Priority:

Severity is the **impact** a defect has on the software's functionality, while priority is the **urgency** to fix it. Severity is an objective measure assigned by a tester to determine how serious the bug is, whereas priority is a subjective measure decided by a manager or client to determine the order of fixes based on business needs.

For example, a bug that **crashes** the application is **high** severity, but it could be **low** priority if it only happens in a rarely used, non-critical feature

Severity

Definition: The degree to which a defect impacts the software's functionality and user experience.

Assigner: Typically a tester.

Measurement: An objective measure, often categorized as:

Critical: The application crashes or data is lost.

Major: A major feature is blocked.

Moderate: A feature is partially impaired.

Minor: A minor inconvenience that is a workaround.

Cosmetic: A visual issue with no functional impact.

Priority

Definition: The order in which defects should be fixed based on their importance to the business.

Assigner: Typically a manager or client, often in consultation with the testing team.

Measurement: A subjective measure, often categorized as:

High: Fix immediately.

Medium: Fix in the next release.

Low: Fix when time permits.

Examples of Priority and Severity Combination

There are possible combinations of priority and severity:

1. High Severity High Priority: Consider an example of a web application where the if after filling in the login details, the user cannot click the login button then this is the case of high severity and high priority.

High Severity: If the login button is not clickable then the whole application is blocked and none of the functions can be accessed by the user

High Priority: If the login button is not clickable this means that the application is not letting any user log in then what is the use of such an application? Such defects are high-priority defects as the users will avoid such applications and businesses will be impacted.

2. High Severity Low Priority: Consider the example of the application being used on the older versions of Internet Explorer say IE8. This is a case of high severity and low priority.

High Severity: The fault, in this case, is of high severity because when the application is accessed on the older version, the page will not load properly and a few fields and text will be overlapped thus whole application will be impacted.

Low Priority: The defect is of low priority because very few actual users use IE8 or older versions so the fix can wait.

3. Low Severity Low Priority: Consider the example of the help or faq section of the website where the theme or font style of a section of the page does not match with that of the rest of the page.

Low Severity: The defect is of low severity as the defect is not affecting the website functionality.

Low Priority: The defect is of low priority as not many users will access this particular section of the website so the fix can wait.

4. Low Severity High Priority: Consider the example when there is a typo on the website. For example, the case of the school website where the 'Admission Form' is misspelled as 'Admission from'.

Low Severity: The defect doesn’t break functionality-the page and form still work; it’s a UI/text issue.

High Priority: It’s business-critical and user-facing (core CTA), hurts credibility and conversions (admissions), so it must be fixed immediately.