**Bug Life Cycle**

1. **What is the difference between Error, Defect, Bug and failure?**

**Ans.**

1. Error: - Something is missing by the developer or testing in requirement
2. Defect: - During the testing tester the difference found by the tester between Actual and expected is call defect.
3. Bug: - once the developer is accepted the reported defect then it is called Bug.
4. Failure: - the major mistake by the developer and tester that would found by the end user or client.
5. **What is the difference between Latent and masked Defect?**

**Ans.** Latent defects is an existing defect that has not yet caused a failure just because the exact set of conditions has never been met.

Masked defect is an existing defect that hasn’t yet caused a failure just because another defect has prevented that part of the code from being executed.

1. **How Do You Handle A Non-Reproducible Bug?**

Answer. Following bugs lie under the non-reproducible category.

1. The race condition is an error scenario which occurs when the timing of one event impacts another executing in a sequence.
2. Defects observed due to low memory issue.
3. Bugs raised due to address pointing to a memory location that does not exist.

A tester can take the following actions to handle the non-reproducible bugs.

1. Keep the resources & time constraints under check.
2. Execute test steps that are close to the error description.
3. Evaluate the test environment.
4. Examine and evaluate test execution results.
5. **What Will You Do When A Bug Turns Up During Testing?**

Answer. When a bug shows up, we can follow the below steps.

Run more tests to make sure that the problem has a clear description.

Run a few more tests to ensure that the same problem doesn’t exist with different inputs.

Once we are sure of the full scope of the bug, then we can add details and report it.

1. **What is the difference between Priority and severity?**

Ans. The Word Priority and severity are used in bug Tracking to share the importance of a bug among the team to fix it

Priority: - IS found as user point of view .

Severity: - Is found in the application point of view.

1. **Give an example of High priority and low severity?**

**Ans.** No error message to prevent Wrong operation in application.

1. **Give an example of High priority and High severity?**

**Ans.** Application does not allow multiple user while it required in requirement.

1. **Give an example of Low priority and High severity?**

**Ans.**  Application does not allow customer expected configuration.

1. **Give an example of low priority and low severity?**

**Ans. E**rror message have complex meaning

1. **What Are Different Types Of Severity?**

Answer. The severity of a bug can be low, medium or high depending on the context.

1. User Interface Defect - Low
2. Boundary Related Defects - Medium
3. Error Handling Defects - Medium
4. Calculation Defects - High
5. Misinterpreted Data - High
6. Hardware Failures - High
7. Compatibility Issues - High
8. Control Flow Defects - High
9. Load Conditions (Memory leakages under load testing) – High
10. **What Kind Of Document Will You Need To Begin Functional Testing?**

Answer.

It is none other than the Functional specification document. It defines the full functionality of a product.

Other documents are also useful in testing like user manual and BRS.

Gap analysis is another document which can help in understanding the expected and existing system.

1. **What Is The Primary Difference Between Debugging & Testing?**

Answer. Testing is to find out defects while using a product whereas debugging is to reach the part of the code causing failure.

Debugging is isolating the problem area in the code done by a developer whereas Testing is identifying the bug in an application and done by a tester.

1. **What Are Error Guessing And Error Seeding?**

Answer.

Error Guessing.

It is a test case design technique in which testers have to guess the defects that might occur and write test cases to represent them.

Error Seeding.

It is the process of adding known bugs in a program for the tracking the rate of detection & removal. It also helps to estimate the number of faults remaining in the program.

1. **Why Does Software Have Bugs?**

Answer.

1. Miscommunication.
2. Programming errors.
3. Timeline pressures.
4. Change in requirements.
5. Software complexity
6. **What is Error?**

Ans: Error is mistake due to human action leading incorrect result.

Example: A developer may misunderstand a de-sign notation, or a programmer might type a variable name incorrectly – leads to an Error.

1. **What is Fault?**

Ans:

1. **What is failure?**

Ans: A failure is the inability of a software system or component to perform its required functions within specified performance requirements.

1. **What is bug/Defect?**

Ans: Bug/Defect indicate difference between expected behaviour and actual behaviour.

1. **How do you report defect?**

Ans: Defect are reported using a defect management tool loke Bugzilla. When one finds a bug in the software, he enters the detail realated to the defect like module name, description, type, priority and severity etc. From these, It is tracked to completion

**Note:**  For smaller project, microsoft excel worksheet is used for tracking defect.

1. **Why are defect report prepared?**

Ans: Defect report is a complete record of discrepancies. It is prepared so that it can be used in multiple ways to improve quality.

* To get the defect fixed.
* To report status of the application.
* To generate statistics to predict defect
* To arrive at defect prevention plan
* For driving process improvement

1. **What are the content of a defect report?**

Ans: Defect report is a complete record of discrepancies. Following are the attribute used in a defect report.

|  |  |
| --- | --- |
| Defect Id | For easier indetificationand cross referencing in document like test logs |
| Project Name | Project name under test |
| Module Name | Module under test |
| Sub-module name | Sub-module under test |
| Phase | SDLC phase |
| Type | Functional/performance/Security/UI |
| Severity | Very high,High,Medium,Low |
| Prioprity | Very high,High,Medium,Low |
| Summary | Brief Description |
| Description | Detail: Step to reproduce,HW/SW configuration,Screen shot etc |
| Status | New,opened,Assigned,fixed,duplicate,deferred,closed |
| Reported by | Tester name and date |
| Assigned To | Name of developer to whom it is assign for fixing |

1. **What is important,severity or priority while fixing a defect?**

Ans: Severity and priority both are important for fixing the defect, as their combination decides the sequence in which the defect would get fixed. All ‘High severity and high prioprity’, ‘High prioprity and low severity’ and ‘High severity and low priority’ would need immediate attention.

1. **Suppose you found some bugs with low severity which may not affect the application and the client needs release today. What will be yours action? Why?**

Ans: If the bugs have low severity and you think they are also not of high priority then the release can go as it is without fixing thise bugs. If some of them are of high priority, you may bring this to the notice of your lead and he can take subcequent action.

1. **Who will decide severity? Developer or Tester?**

Ans: Software tester would assign severity for a defect.

1. **In defect life cycle,you find that the defect is not fixed but the developer saya that it is fixed. Will you tell the project lead every time about it?**

Ans: As a process , software tester does not directly come in contact with the developer to convience him that the bugs are not fixed. By the process, When a tester reports a bug and is fixed by the developer, developer would change the status of the defect as fixed. Software tester has to re-test and ascertain that defect is really fixed and then close it. Till software tester close the defect, it remain open on a developer’s name and he has the responsibility to fix it. If the defect reporting is clear and concise then there is nothing a tester to do furthur.

1. **How will you convience a developer that it is a defect if he is does not agree?**

Ans: Firstly, convincing a defect or non-defect is not an emotional issue. Defect is deviation from the expected behaviour as stated in the baseline document of SRS(Software Requirement Specification).As a process , software tester does not directly come in contact with the developer to convience him that the bugs are not fixed. If the following precaution are taken while reporting the defect, the problem would get automatically addressed.

* Details on the defect like module, location etc are entered completely and correctly.
* Steps to reproduce the defect are listed in an unambiguous way.
* Any evidence like a screen shot is a attached with the defect.
* If tester can illustrate kind of impact the defect would have on the application, the defect would get the desire attention.

1. **Does every critical bugs have high priority?**

Ans: Generally yes, However there can be exception. There is a critical bug in a functionality which is not frequently being used and probability of occurance is also not so high then, considering smaller risk impact of this bug it may not be given high priority. The decision is taken in the context of the application and the risk perceived.

1. **What is defect density?**

Ans: Defect density indicates quality of code developed. It is measure of number of defect per unit size of software. This unit size could be expressed at KLOC(Kilo lines of code).

Defect density = Total number of defect /Size \*100

1. **What is defect leakage/defect removal efficiency/defect detection efficiency?**

Ans: This metrics measure defect leaked to the production environment as a percentage of total defect logged by testing team. This measure efficiency of testing team.

Defect leakage = Total defect found in production/(Total defect in house+production)\*100%

1. **What is defect age?**

Ans: Defect age is the difference in time between the day when the defect was found and the time when it was closed or current day. It is a measure of responsiveness of development team to fix the bugs.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***