Test Management

JAN, 2025

expleo

Test Management

Overview

Test Organization

Test Planning and Estimation

Test Monitoring and Control

Configuration Management

Defect Management

Test Organization

Test Organization and Independence:

- A certain degree of independence often makes the tester more effective at finding defects and failures.
- Degrees of independence in testing include the following (from low level of independence to high level):
  - No independent testers; the only form of testing available is developers testing their own code
  - Independent developers or testers within the development teams or the project team; this could be developers testing their colleagues' products
  - Independent test team or group within the organization, reporting to project management or executive management

Test Organization

- Independent testers from the business organization or user community, or with specializations in specific test types such as usability, security, performance, regulatory/compliance, or portability
- Independent testers external to the organization, either working on-site (in-house) or off-site (outsourcing)

Test Organization

Test Organization and Independence:

- **Potential benefits of test independence include:**
  - Independent testers are likely to recognize different kinds of failures compared to developers because of their different backgrounds, technical perspectives, and biases
  - An independent tester can verify, challenge, or disprove assumptions made by stakeholders during specification and implementation of the system
  - Independent testers of a vendor can report in an upright and objective manner about the system under test without (political) pressure of the company that hired them Independent test team or group within the organization, reporting to project management or executive management

Test Organization

Test Organization and Independence:

- Potential drawbacks of test independence include:

➢ Isolation from the development team, may lead to a lack of collaboration, delays in providing feedback to the development team, or an adversarial relationship with the development team

➢ Developers may lose a sense of responsibility for quality

➢ Independent testers may be seen as a bottleneck

➢ Independent testers may lack some important information (e.g., about the test object)

Test Organization

Task of Test Manager

- The test manager is tasked with overall responsibility for the test process and successful leadership of the test activities.
- In larger projects or organizations, several test teams may report to a test manager, test coach, or test coordinator, each team being headed by a test leader or lead tester.
- Typical Tasks includes:
  - Develop or review a test policy and test strategy for the organization
  - Plan the test activities by considering the context, and understanding the test objectives and risks. This may include selecting test approaches, estimating test time, effort and cost, acquiring resources, defining test levels and test cycles, and planning defect management

Test Organization

Task of Test Manager

- Write and update the test plan(s)
- Coordinate the test plan(s) with project managers, product owners, and others
- Share testing perspectives with other project activities, such as integration planning
- Initiate the analysis, design, implementation, and execution of tests, monitor test progress and results, and check the status of exit criteria
- Prepare and deliver test progress reports and test summary reports based on the information gathered

Test Organization

Task of Test Manager

- Support the selection and implementation of tools to support the test process, including recommending the budget for tool selection (and possibly purchase and/or support), allocating time and effort for pilot projects, and providing continuing support in the use of the tool(s)
- Decide about the implementation of test environment(s)
- Promote and advocate the testers, the test team, and the test profession within the organization

Test Organization

Task of Tester

- Review and contribute to test plans
- Analyze, review, and assess requirements, user stories and acceptance criteria, specifications, and models for testability (i.e., the test basis)
- Identify and document test conditions, and capture traceability between test cases, test conditions, and the test basis
- Design, set up, and verify test environment(s), often coordinating with system administration and network management
- Design and implement test cases and test procedures

Test Organization

- Prepare and acquire test data
- Create the detailed test execution schedule
- Execute tests, evaluate the results, and document deviations from expected results
- Use appropriate tools to facilitate the test process
- Automate tests as needed (may be supported by a developer or a test automation expert)
- Support the selection and implementation of tools to support the test process, including recommending the budget for tool selection (and possibly purchase and/or support), allocating time and effort for pilot projects, and providing continuing support in the use of the tool(s) nization

Test Organization

Task of Tester

- Create the detailed test execution schedule
- Execute tests, evaluate the results, and document deviations from expected results
- Use appropriate tools to facilitate the test process
- Automate tests as needed (may be supported by a developer or a test automation expert)

Test Planning and Estimation

Purpose of Test Plan:

- A test plan outlines test activities for development and maintenance projects. Planning is influenced by the test policy and test strategy of the organization, the development lifecycles and methods, the scope of testing, objectives, risks, constraints, criticality, testability, and the availability of resources.
- Test Planning Activities:
  - Determining the scope, objectives, and risks of testing
  - Defining the overall approach of testing
  - Integrating and coordinating the test activities into the software lifecycle activities
  - Making decisions about what to test, the people and other resources required to perform the various test activities, and how test activities will be carried out

Test Planning and Estimation

- Test Planning Activities:

➢Scheduling of test analysis, design, implementation, execution, and evaluation activities, either on particular dates (e.g., in sequential development) or in the context of each iteration (e.g., in iterative development)

➢Selecting metrics for test monitoring and control

➢Budgeting for the test activities

➢Determining the level of detail and structure for test documentation (e.g., by providing templates or example documents)

Test Planning and Estimation

Typical Entry Criteria

- Availability of testable requirements, user stories, and/or models (e.g., when following a modelbased testing strategy)
- Availability of test items that have met the exit criteria for any prior test levels
- Availability of test environment
- Availability of necessary test tools
- Availability of test data and other necessary resources

Test Planning and Estimation

Typical Exit Criteria

- Planned tests have been executed
- A defined level of coverage (e.g., of requirements, user stories, acceptance criteria, risks, code) has been achieved
- The number of unresolved defects is within an agreed limit
- The number of estimated remaining defects is sufficiently low
- The evaluated levels of reliability, performance efficiency, usability, security, and other relevant quality characteristics are sufficient

Test Monitoring and Control

Test Monitoring

- The purpose of test monitoring is to gather information and provide feedback and visibility about test activities.
- Information to be monitored may be collected manually or automatically and should be used to assess test progress and to measure whether the test exit criteria.

Test Control

- Test control describes any guiding or corrective actions taken as a result of information and metrics gathered and (possibly) reported.
- Actions may cover any test activity and may affect any other software lifecycle activity.

Test Monitoring and Control

Test Control

- Examples of test control actions include:
- Re-prioritizing tests when an identified risk occurs (e.g., software delivered late)
- Changing the test schedule due to availability or unavailability of a test environment or other resources
- Re-evaluating whether a test item meets an entry or exit criterion due to rework

Test Monitoring and Control

Metrics Used in Testing

- Metrics can be collected during and at the end of test activities in order to assess:

➢ Progress against the planned schedule and budget

➢ Current quality of the test object

➢ Adequacy of the test approach

➢ Effectiveness of the test activities with respect to the objectives

Test Monitoring and Control

Common test metrics include:

- Percentage of planned work done in test case preparation (or percentage of planned test cases implemented)
- Percentage of planned work done in test environment preparation
- Test case execution (e.g., number of test cases run/not run, test cases passed/failed, and/or test conditions passed/failed)
- Defect information (e.g., defect density, defects found and fixed, failure rate, and confirmation test results)
- Test coverage of requirements, user stories, acceptance criteria, risks, or code
- Task completion, resource allocation and usage, and effort

Test Monitoring and Control

Typical test summary reports may include:

- Summary of testing performed
- Information on what occurred during a test period
- Deviations from plan, including deviations in schedule, duration, or effort of test activities
- Status of testing and product quality with respect to the exit criteria or definition of done
- Factors that have blocked or continue to block progress
- Metrics of defects, test cases, test coverage, activity progress, and resource consumption.
- Reusable test work products produced

Configuration Management

- The purpose of configuration management is to establish and maintain the integrity of the component or system, the testware, and their relationships to one another through the project and product lifecycle.
- To properly support testing, configuration management may involve ensuring the following:

➢All test items are uniquely identified, version controlled, tracked for changes, and related to each other

➢All items of testware are uniquely identified, version controlled, tracked for changes, related to each other and related to versions of the test item(s) so that traceability can be maintained throughout the test process

➢All identified documents and software items are referenced unambiguously in test documentation Reusable test work products produced

Quiz

1) From a Testing perspective, what are the MAIN purposes of Configuration Management?:

i) Identifying the version of software under test.
ii) Controlling the version of testware items.
iii) Developing new testware items.
iv) Tracking changes to testware items.
v) Analysing the need for new testware items.

a) ii, iv, and v

b) i, ii, iii and iv

c) i, ii, and iv

d) i, iii and iv

Answer : Option c)

Quiz

2) Which of the following is a MAJOR task of test planning?

a) Scheduling test analysis and design tasks.

b) Initiating corrective actions.

c) Monitoring progress and test coverage.

d) Measuring and analyzing results.

Answer : Option a)

Quiz

3) As part of which test process do you determine the exit criteria?

a) Test planning.

b) Evaluating exit criteria and reporting.

c) Test closure.

d) Measuring and analyzing results.

Answer : Option a)

Quiz

4) Which of the following is a benefit of test independence?

a) It does not require familiarity with the code

b) It is cheaper than using developers to test their own code.

c) It avoids author bias in defining effective tests.

d) Testers are better at finding defects than developers.

Answer : Option c)

Quiz

6) A project that is in the implementation phase is six weeks behind schedule. The delivery date for the product is four months away. The project is not allowed to slip the delivery date or compromise on the quality standards established for his product. Which of the following actions would bring this project back on schedule?

a) Eliminate some of the requirements that have not yet been implemented.

b) Add more engineers to the project to make up for lost work.

c) Ask the current developers to work overtime until the lost work is recovered.

d) Hire more software quality assurance personnel.

Answer : Option a)

Quiz

8) Which of the following is the task of a Tester?

i. Interaction with the Test Tool Vendor to identify best ways to leverage test tool on the project.
ii. Prepare and acquire Test Data
iii. Implement Tests on all test levels, execute and log the tests.
iv. Create the Test Specifications

a) i, ii, iii is true and iv is false

b) ii,iii,iv is true and i is false.

c) i is true and ii,iii,iv are false

d) iii and iv is correct and i and ii are incorrect

Answer : Option b)

Quiz

9) Which of the following helps in monitoring the Test Progress:

i. Percentage of Test Case Execution
ii. Percentage of work done in test environment preparation.
iii. Defect Information e.g. defect density, defects found and fixed
iv. The size of the testing Team and skills of the engineers

a) i, ii, iii is true and iv is false

b) ii,iii,iv is true and i is false.

c) i is true and ii,iii,iv are false

d) iii and iv is correct and i and ii are incorrect

Answer : Option a)

Quiz

10) Which of the following is the task of a Test Lead / Leader.

i. Interaction with the Test Tool Vendor to identify best ways to leverage test tool on the project.
ii. Write Test Summary Reports based on the information gathered during testing
iii. Decide what should be automated , to what degree and how.
iv. Create the Test Specifications

a) i, ii, iii is true and iv is false

b) ii,iii,iv is true and i is false.

c) i is true and ii,iii,iv are false

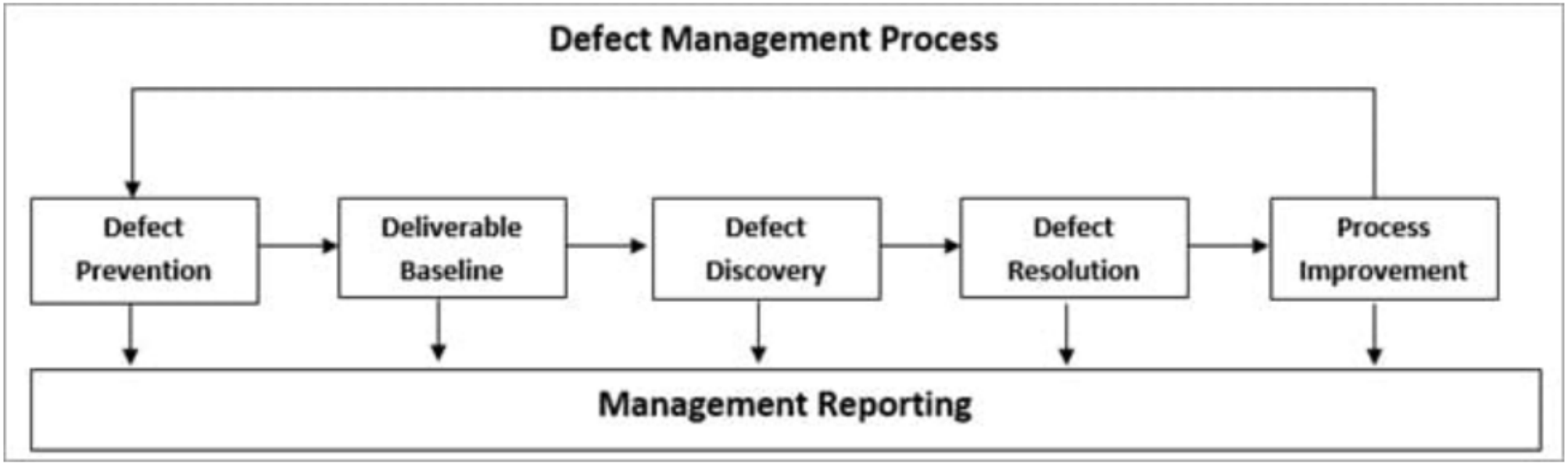
d) iii and iv is correct and i and ii are incorrect

Answer : Option a)

Defect/Bug Life Cycle

Defect Management

Defect Management Process



Defect Management

Defect Management Process

1. Defect Prevention:

- Defect Prevention is the best method to eliminate the defects in the early stage of testing instead of finding the defects in the later stage and then fixing it. This method is also cost effective as the cost required for fixing the defects found in the early stages of testing is very low.
- The major steps involved in Defect Prevention are as follow:
  - Identify Critical Risk:
  - Estimate Expected Impact:
  - Minimize expected impact:

Defect Management

2. Deliverable Baseline:

- When a deliverable (system, product or document) reaches its pre-defined milestone then you can say a deliverable is a baseline.
- For Example, consider a scenario of coding, unit testing and then system testing.
  - If a developer performs coding and unit testing then system testing is carried out by the testing team. Here coding and Unit Testing is one milestone and System Testing is another milestone.
  - So during unit testing, if the developer finds some issues then it is not called as a defect as these issues are identified before the meeting of the milestone deadline. Once the coding and unit testing have been completed, the developer hand-overs the code for system testing and then you can say that the code is “baselined” and ready for next milestone, here, in this case, it is “system testing”.

Defect Management

3. Defect Discovery:

- It is almost impossible to remove all the defects from the system and make a system as a defect-free one. But you can identify the defects early before they become costlier to the project.
- Steps involved in Defect Discovery are as follows:
  - Find a Defect
  - Report Defect
  - Acknowledge Defect

Defect Management

4. Defect Resolution:

- The testing team has identified the defect and reported to the development team. Now here the development team needs to proceed for the resolution of the defect.
- Steps involved in Defect Resolution are as follows:
  - Prioritize the risk
  - Fix the defect
  - Report the resolution

Defect Management

5. Process Improvement:

- Though in the defect resolution process the defects are prioritized and fixed, from a process perspective, it does not mean that lower priority defects are not important and are not impacting much to the system. From process improvement point of view, all defects identified are same as a critical defect.
- For process improvement, everyone in the project needs to look back and check from where the defect was originated. Based on that you can make changes in the validation process, base-lining document, review process which may catch the defects early in the process which are less expensive.

Defect Management

- Typical defect reports have the following objectives:

➢ Provide developers and other parties with information about any adverse event that occurred, to enable them to identify specific effects, to isolate the problem with a minimal reproducing test, and to correct the potential defect(s), as needed or to otherwise resolve the problem

➢ Provide test managers a means of tracking the quality of the work product and the impact on the testing (e.g., if a lot of defects are reported, the testers will have spent a lot of time reporting them instead of running tests, and there will be more confirmation testing needed)

➢ Provide ideas for development and test process improvement

Bug / Defect

- A bug/defect is an issue or error in a software application that prevents it from functioning as intended.
- It typically arises due to programming or design errors made during development.
- Bugs are usually detected during testing by testers or reported by users.
- The tester’s responsibility includes:
    - Detecting and reporting bugs.
    - Ensuring bugs are fixed promptly.
    - Verifying that the bug fixes result in a quality, error-free application.

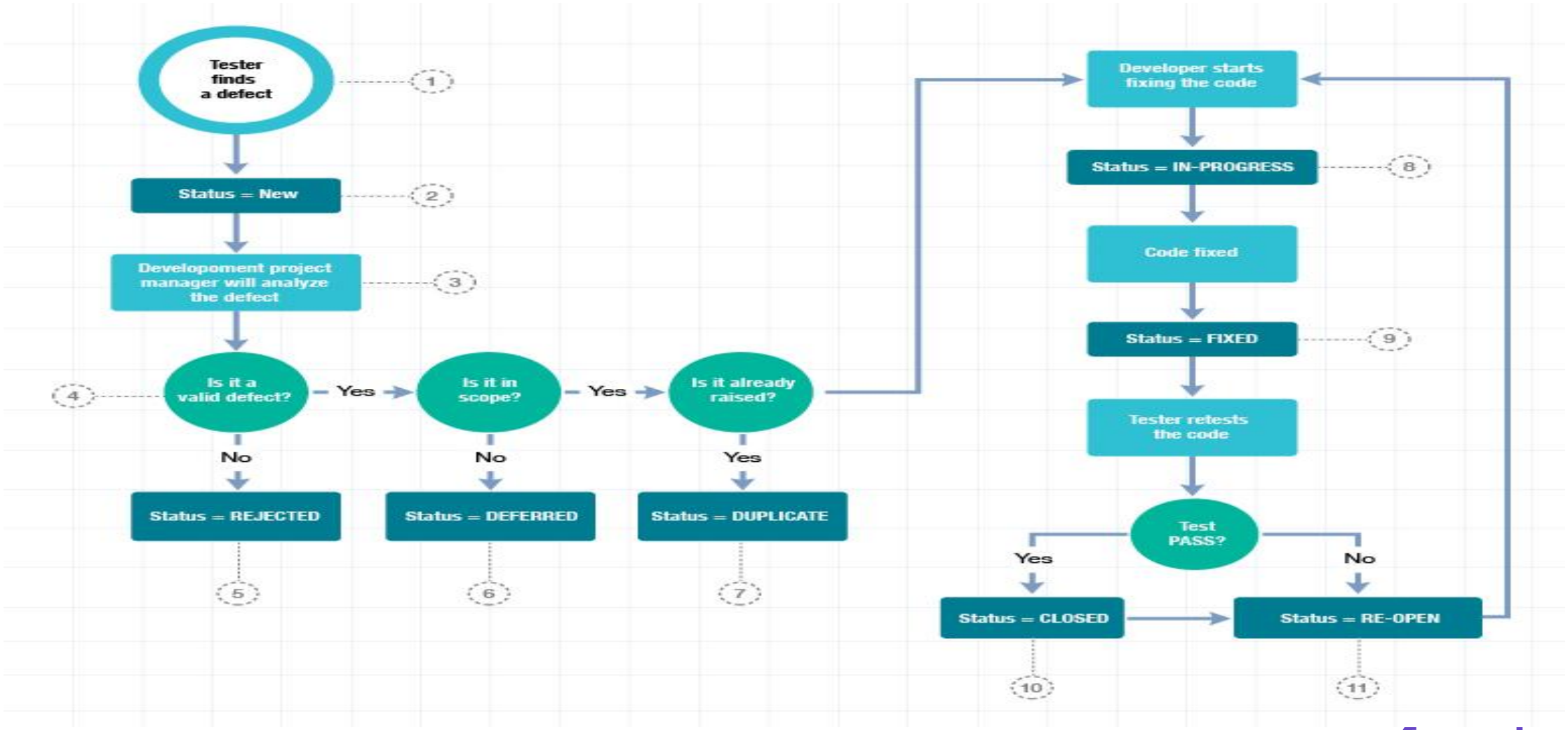
Bug Life Cycle (Defect Life Cycle)

- The Bug Life Cycle is the journey of a defect from identification to closure.
- It helps track and manage the status of bugs throughout the software development process.
- Different team members (testers, developers, project managers) are involved at various stages.

Defect/Bug Status:

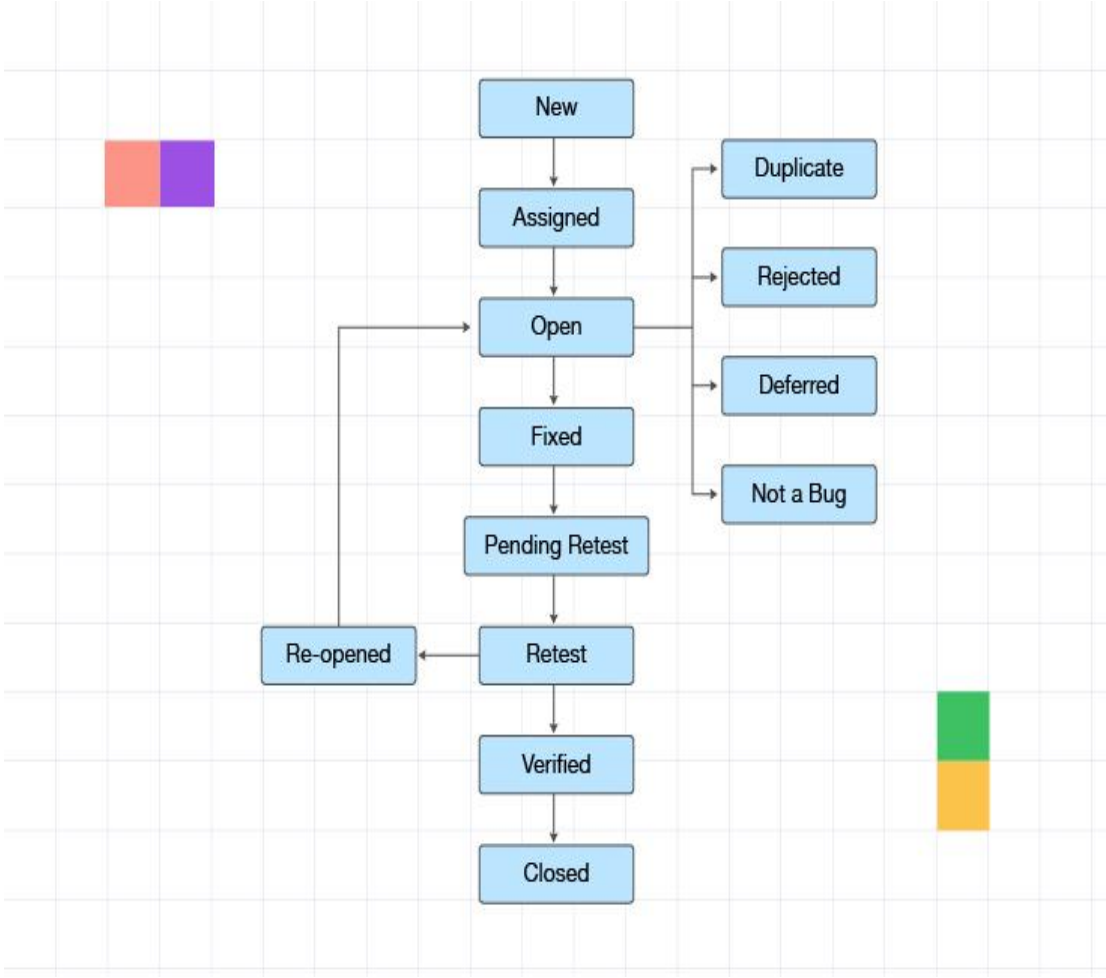
- Defect Status refers to the current state of a defect in its life cycle.
- It provides visibility into the progress and handling of the bug.

Bug Life Cycle (Defect Life Cycle)



Stages of Bug Life Cycle

- The number of stages that a bug/defect passes through in the bug life cycle in software testing depends on the type of your project.
- The following section familiarizes you with all stages of the bug life cycle.



Stages of Bug Life Cycle

Stage	Description
1. New	A new defect is identified and logged by the tester.
2. Assigned	The defect is assigned to a developer (either directly or via a Dev Lead).
3. Open	The developer starts analyzing and working on the defect.
4. Fixed	Developer resolves the defect and marks it as "Fixed".
5. Test	The defect is ready for testing to verify the fix.
6. Verified	Tester confirms the defect is resolved by re-testing it.
7. Closed	The bug is verified and marked as closed.
8. Reopen	Bug reappears or is not fixed properly—reopened for further action.
9. Duplicate	The defect has already been reported—marked as duplicate.
10. Deferred	Fix postponed to a future release (usually for low-priority issues).
11. Rejected	The bug is considered invalid or irrelevant.
12. Cannot be fixed	Due to constraints (tech, cost, etc.), bug cannot be resolved.
13. Not a defect	The reported issue doesn't affect functionality or isn’t a defect.
14. Not Reproducible	The issue can't be replicated due to lack of information or mismatch in environment.
15. Need More Information	Developer requires more data to understand or reproduce the issue.

Stages of Bug Life Cycle

1. New:

- Whenever a new defect is registered and posted for the first time, a “NEW” status is assigned to it.
- The tester offers a detailed defect/bug report to the Development team to let the team refer to the document and resolve the bug accordingly.

2. Assigned:

- After the tester posts the bug, the tester’s lead authorises the bug and assigns it to the developer team. In this stage, there can be two conditions.
- Firstly, the defect can be directly assigned to the developer.
- Secondly, it can be assigned to the Dev Lead and after being approved by the Dev Lead, they can further transfer the defect to the developer.

Stages of Bug Life Cycle

3. Open:

- In this stage, the developer begins to analyse and resolve the defect. Primarily, the stage involves a developer team that detects and resolves errors in testing and code to ensure that the mistakes have been rectified.
- If the developer team perceives that the defect is inappropriate, it is moved to either the 'Deferred' or the 'Rejected' state.

4. Fixed:

- A developer assigns a "Fixed" status to a bug after making a relevant code change and verifying the change. At this stage, the Dev Lead is informed that the defects available on the Fixed status are the defects that would be available to a tester for testing in the upcoming build.

Stages of Bug Life Cycle

5. Re-Test:

- The bug status as “Test” indicates that the defect is fixed and is ready to perform the test. The test is done to verify that the bug is fixed.

6. Verified:

- In this stage, the tester re-tests the bug after being fixed by the developer. In case the bug is not reproduced, the bug is considered fixed, and a “verified” status is assigned to it.

7. Closed:

- If the bug is resolved, the tester allocates the “Closed” status to it. After the bug is fixed, the tester tests it. This stage indicates that the bug is fixed, tested, and validated.

Stages of Bug Life Cycle

8. Reopen:

- In certain scenarios, the bug exists even after the developer has resolved it.
- In those cases, the tester assigns a “Reopen” state to the bug.
- Subsequently, the bug passes through the same life cycle again after the “Reopen” bug is moved to “fixed” state.

9. Duplicate:

- Sometime, the same bug may be reported twice.
- In that case, the bug status is modified to “duplicate.” Subsequently, the defect is rejected.

Stages of Bug Life Cycle

10. Deferred:

- The tester/bug triage team assigns a "Deferred" state to bugs in the following cases:

- If the bug is not of key priority and is expected to be resolved in the forthcoming release.

- The bug is expected to be resolved in the forthcoming release.

- The customer intends to modify the requirement

Note that the high-priority bugs can't and should not be deferred.

11. Rejected:

- If the developer perceives that the defect is due to misinterpretation or is not genuine, they will assign a "Rejected" state to the defect. Usually, the reason for rejection may be one of the following: **NOT a Defect, Duplicate Defect and Non-Reproducible**

Stages of Bug Life Cycle

12. Cannot be fixed:

- The developer assigns “Can’t be fixed” state to the defect under the following cases:
  - Unsupported technology
  - High cost of fixing bug
  - Lack of necessary skills

13. Not a defect:

- If the defect doesn’t impact other functions of the software then it is assigned a ‘NOT A DEFECT’ state. Ultimately, it is ‘Rejected’.

Stages of Bug Life Cycle

14. Not Reproducible:

- The developer assigns a “Not Reproducible” state to the defect if it can’t be reproduced due to:
    - Inappropriate defect document
    - Platform mismatch
    - Data mismatch
    - Build mismatch
    - Inconsistent defects

Stages of Bug Life Cycle

15. Need more information:

- The developer assigns the “Need more information” status to the defect if they can’t reproduce the defect according to the guidelines defined by a tester.
- At this stage, the tester must add elaborated reproducing steps and also assign bug back to the development team for its solution.
- The tester can avoid these hassles by writing a comprehensive defect document.

Example of a Bug Life Cycle

Scenario: Bug in Flight Booking Software

Bug: Meal not booked with ticket.

Tester logs the bug: Status = **New**

Project Manager reviews and assigns: Status = **Assigned**

Developer fixes: Status = **Fixed**

Tester tests, but finds pricing missing: Status = **Reopened**

Developer re-fixes and tester re-verifies: Status = **Verified**

Tester confirms everything works: Status = **Closed**

Defect Triage Meeting

- **Defect triage** is a process where **each bug is prioritized based on its severity, frequency, risk, etc.**
- Triage term is used in the Software testing / QA to define the severity and priority of new defects.
- The goal of Bug Triage is to evaluate, prioritize and assign the resolution of defects.
- The team needs to validate severities of the defect, make changes as per need, finalize resolution of the defects, and assign resources.
- Mainly used in agile project management.

Defect Triage Meeting - Participants

Mandatory Participants: Below project members always take part in Defect Triage Meetings.

- Project Manager
- Test Team Leader
- Technical Lead
- Development Team Leader

Optional Participants:

- Developers
- Testers
- Business Analyst

What happens during 'Defect Triage' Meeting?

- Test Team leader sends out a bug report with the new defects. During the defect triage meeting, each defect is analyzed to see whether right priority and severity are assigned to it.
- Priorities are rearranged if needed.
- Defects are analyzed and evaluated by the degree of their severity.
- This include discussion regarding the complexity of the defect, risks, rejection, reassignment of errors is done.
- Updates are captured in bug tracking system.
- The QA engineer will make the changes to each defect and discuss them with each attendee.
- The "Comments" field is updated correctly by noting essential points of the meeting.

Guidelines on deciding the Severity of Bug

- It indicates the impact each defect has on testing efforts or users and administrators of the application under test.
- This information is used by developers and management as the basis for assigning priority of work on defects.
- "Severity" is associated with standards.
- "Severity" is the state or quality of being severe; severe implies adherence to rigorous standards or high principles and often suggests harshness; severe is marked by or requires strict adherence to rigorous standards or high principles, e.g. a severe code of behavior.

Guidelines for assignment of Priority Level

- **"Priority" is associated with scheduling.** "Priority" means something is afforded or deserves prior attention; precedence established by order of importance (or urgency).

1. Critical / Show Stopper :

- An item that prevents further testing of the product or function under test can be classified as a Critical Bug. No workaround is possible for such bugs. Examples of this include a missing menu option or security permission required to access a function under test.

2. Major / High :

- A defect that does not function as expected/designed or that causes other functionality to fail to meet requirements can be classified as a Major Bug. The workaround can be provided for such bugs. Examples of this include inaccurate calculations; the wrong field being updated, etc.

Guidelines for assignment of Priority Level

3. Average / Medium :

- The defects which do not conform to standards and conventions can be classified as Medium Bugs. Easy workarounds exists to achieve functionality objectives.
- Examples include matching visual and text links which lead to different end points.

4. Minor / Low :

- Cosmetic defects which does not affect the functionality of the system can be classified as Minor Bugs.

Bug Report Preparation

- Developers are often under a ton of pressure to solve issues quickly without actually having a lot of time on their hands.They usually face two extremes: too much unhelpful information or too little important information.

1. Title/Bug ID

- Keep it short and specific.Make sure it clearly summarizes what the bug is. Having a clear title on your bug report makes it easier for the developer to find later on and merge any duplicates.

Examples:

❌ Bad: "I can't see the product when I add it, for some reason I try and it doesn't. WHY? Fix it asap."

- vague, aggressive, too verbose, asks for a solution to be implemented

Bug Report Preparation

☑ Good: “CART - New items added to cart do not appear”

- it helps developers instantly locate the issue (CART)

- it focuses on the actual technical problem

- When developers review it, they’ll be able to instantly assess the issue and move on to other elements of the bug report.

Bug Report Preparation

2. Summary

- If your title isn’t enough, you can add a short report summary. And we mean short. In as few words as possible, include when and how the bug occurred. Your title and description may also be used in searches, so make sure you include important keywords.

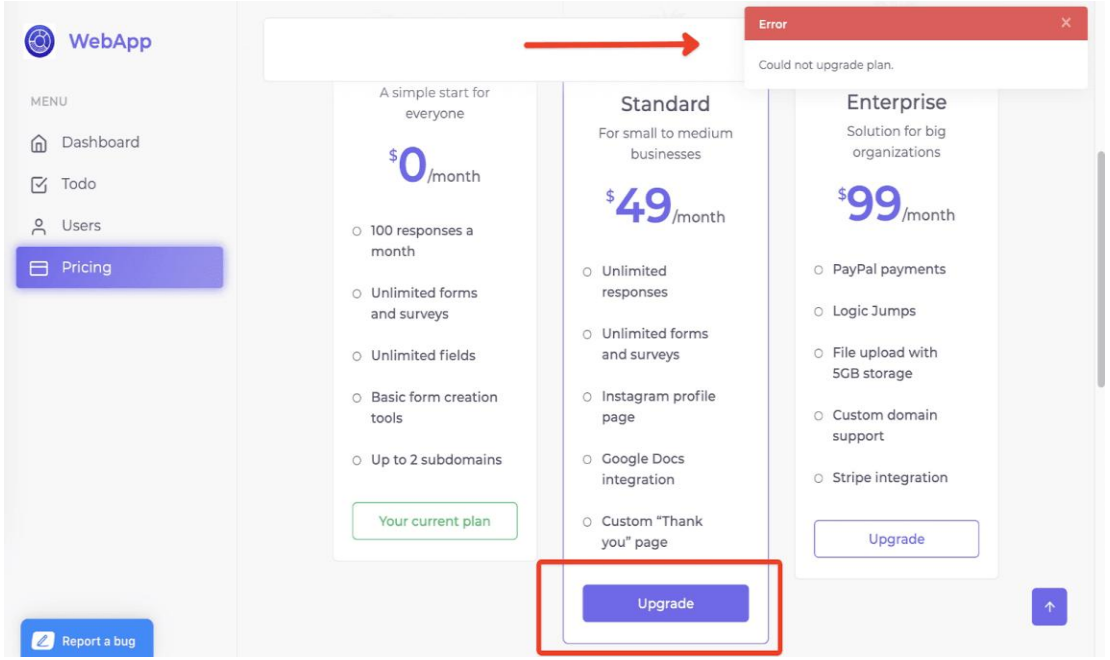
Examples:

- ❌ Bad: "The other day I was trying to add stuff to test and nothing showed up when I did that or clicked on the button."
- ☑ Good: "On [DATE], I tried adding [PRODUCT] to the cart, nothing happened after clicking the 'add' button on the product overview webpage."

Bug Report Preparation

3. Visual proof/screenshot

- We all know that a picture is worth a thousand words. That stands true for bug reporting.
- While it may not tell the whole story, a screenshot or video can add a lot of value by getting your developers to see and understand the problem faster.



Bug Report Preparation

4. Expected vs. actual results

- When you report a bug, take some time to explain to your developer what you expected to happen and what actually happened.

Example:

- Expected result: ”Item should be added to the cart when I click ADD”
- Actual result: ”Item does not appear in the cart”

Bug Report Preparation

5. Steps to reproduce

- Always assume that your developer has no idea about the bug you found—how does he reproduce it? As always, keep it simple!
- The steps to follow should be comprehensive, easy to understand, and short.

Example:

1. Search for product XYZ.
2. Click on product XYZ in search results.
3. Click on “Add to Cart” button.
4. Go to cart.

Bug Report Preparation

6. Environment

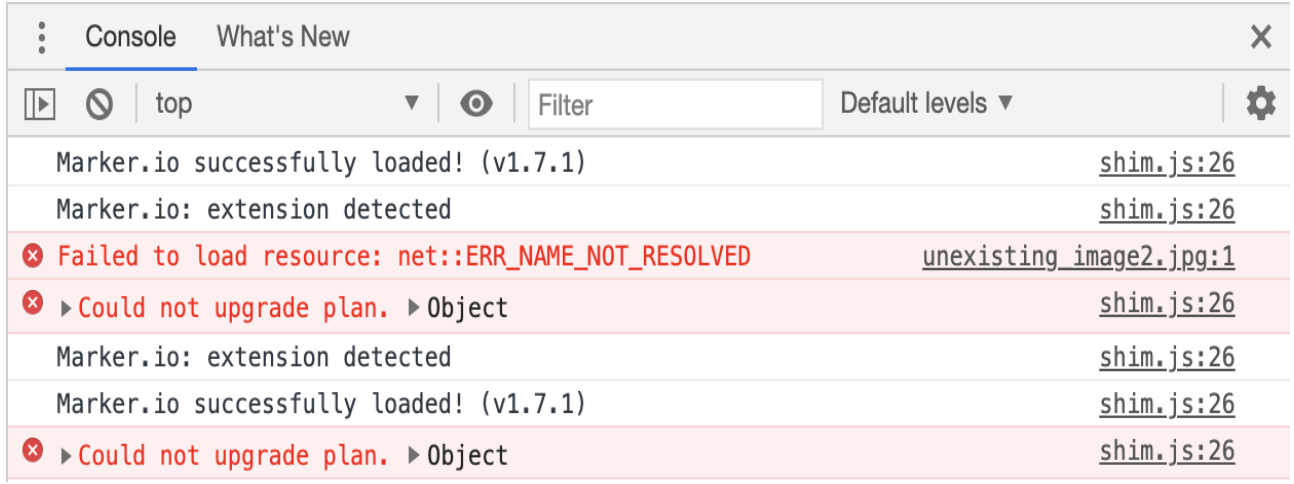
- Websites and apps can behave very differently depending on the environment used.
- It’s critical that the following information be included in any report you share:

Browser	Chrome 88.0.4324.192
Screen Size	1280 x 800
OS	OS X 11.2.1
Viewport Size	1223 x 664
Zoom Level	100%
Pixel Ratio	@2x

Bug Report Preparation

7. Console logs:

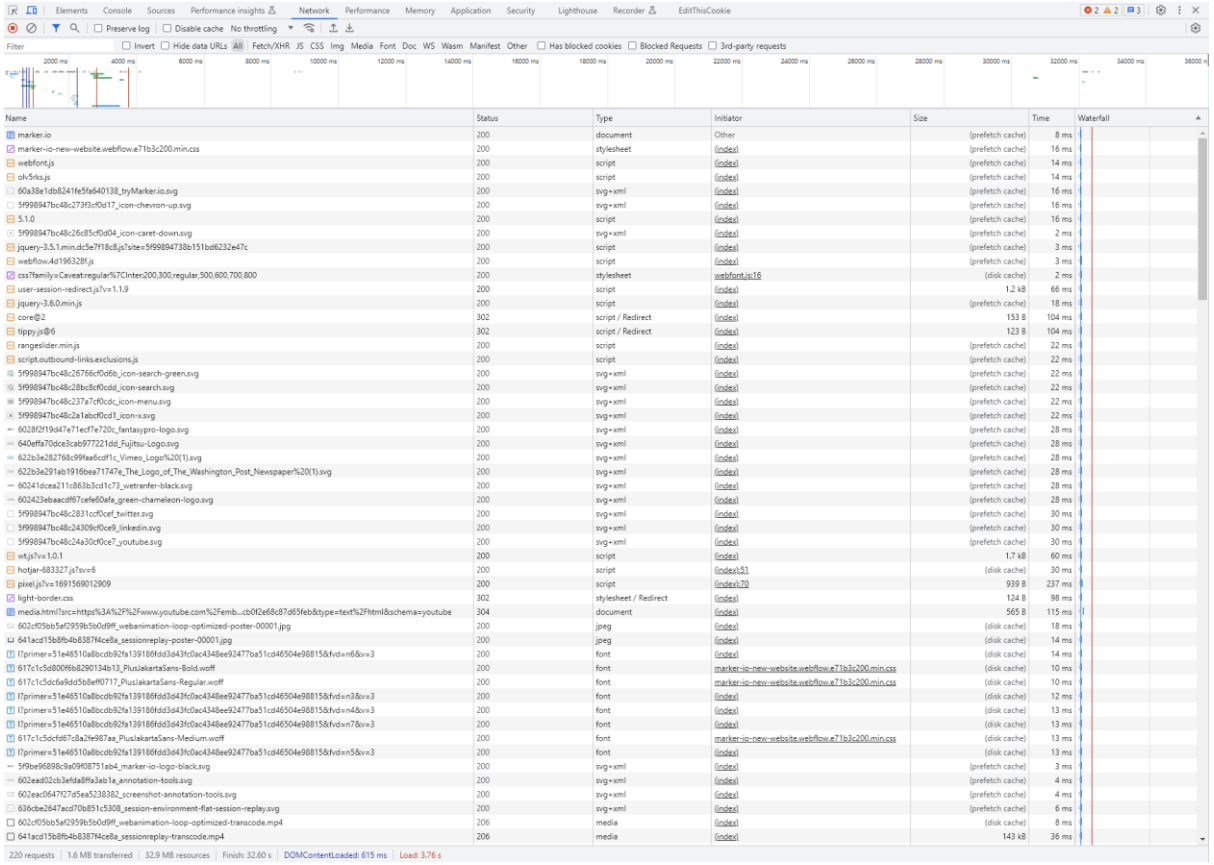
- These logs show developers all errors that occur on a given webpage.
- Logs can also include info that track certain user actions.



Bug Report Preparation

8. Network requests

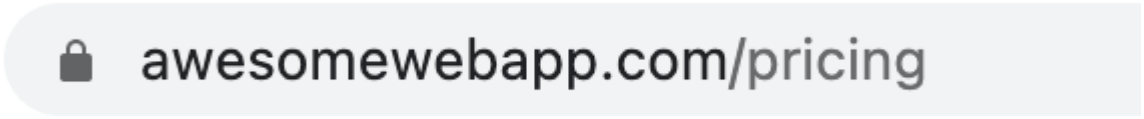
- Network requests detail the flow of data between client and server.
- Inspecting network requests will help developers investigate failed API calls, retrieval of resources (like images), and delayed responses—all things that could slow down a webpage, for example.



Bug Report Preparation

9. Source URL

- One important, but easy-to-forget item is the source URL.
- This will help the developers navigate faster, which saves everyone a lot of time.



- Pro tip: If you use Marker.io, we'll automatically include this, too!

Bug Report Preparation

10. Bug severity and priority

- By defining the severity or priority of the issue in your bug report, your developer understands how quickly a bug should be fixed.Once this has been determined you can label it as:

  - Critical, - Major, - Minor, - Trivial, - Enhancement

- The priority helps your developer determine which bug they should investigate and fix first. Here you can choose between:

  - High

  - Medium

  - Low

Bug Report Preparation

11. Advanced information

- Whether you're looking to write the most informative bug report ever or you're looking to score brownie points with your developer, you can also opt to include the following:

    - Reporter name (your own)
    - Assigned person (the developer, usually)
    - Due date

Quiz

1) Which of the following statements is TRUE about the Bug Life Cycle?

a) A bug is marked as "Closed" as soon as it is assigned to a developer.

b) Severity defines how urgently a bug should be fixed, while priority defines how badly it affects the system.

c) A Defect Triage Meeting is conducted to prioritize and assign bugs for resolution.

d) A bug report only needs the title and the screenshot of the defect.

Answer : Option c)

Quiz

2) In which stage is a bug given to the developer for fixing?

a) Verified

b) Closed

c) Assigned

d) Deferred

Answer : Option c)

Quiz

3) What defines the impact of a defect on the functionality of the application?

a) Priority

b) Complexity

c) Status

d) Severity

Answer : Option d)

Quiz

3) Which stage comes immediately after a bug is fixed?

a) Closed

b) Verified

c) Reassigned

d) Severity

Answer : Option b)