**ROLE OF A QA?**

The role of QA is a very crucial for ensuring the quality, reliability and usability of software products. The primary responsibilities of QA includes:

1. Test planning: understanding requirements, objectives and expected outcomes.
2. Executing necessary testing’s to report defects in the software
3. Collaborating with development team, to ensure that defects are properly identified and resolved.
4. Setting up and maintaining test environments to support testing activities.
5. Continuously monitoring, analyzing and reporting defects if needed.

**What is software testing?**

It is a process that is carried out to evaluate the quality, functionality and performance of a software application. IT IS A METHOD TO CHECK THAT THE ACTUAL SOFTWARE PRODUCT MATCHES EXPECTED REQUIREMENTS. PURPOSE of SOFTWARE TESTING IS TO IDENTIFY ERRORS, GAPS OR MISING REQUIREMENTS IN CONTRAST TO ACTUAL REQUIREMENTS.

**Why is software testing important?**

* To identify defects.by identifying these issues in the development process, testing allows developers to address them before the software is released to users.
* Ensure software quality: testing ensures that the software meets the specified requirements and performs as expected. It helps in validating that all the features and functionalities work correctly and are reliable.
* Validating security: by conducting security testing, it helps to reduce the risk of leakage of important information and can protect sensitive data.
* A properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

**What are the benefits of software testing?**

* It is cost effective: if a bug is caught in an earlier phase, it costs less to fix.
* Security: by identifying and fixing security weaknesses, testing helps to protect sensitive data
* Product quality: testing ensures that a good quality product is delivered to the clients.
* Customer satisfaction: thorough testing results in high and reliable quality of the product. This gives the customer a piece of mind and builds trust and loyalty among users. Satisfied customers are more likely to continue using the software.

**What are the methods of testing?**

**Blackbox testing**: it is a testing technique where internal structure or design are not known or considered. It focuses solely on the inputs and outputs of the system. There is no knowledge of code or algorithms. It is a commonly used test in most projects. UI and UX will be accessible to the tester. No programming skills are required in this testing.

**White box testing**: it is a testing technique that focuses on examining the internal structure and design. This requires a knowledge of the system’s internal code and algorithms of the software. This is carried out by the developers or highly qualified testers. We cannot perform this testing with simple manual testers.

**Gray box testing**: it is a testing technique that combines the elements of black box testing and white box testing.

**SDLC (SOFTWARE DEVELOPMENT LIFE CYCLE)**

It is a methodology that is used to guide the development and maintenance of software systems.it is a framework that is followed by the development team to plan, design develop and deploy in software applications It has 6 phases.

1. **Requirement gathering and Analysis:**

This is done by the Business Analyst. The BA gathers all the requirements from the client. They will then analyze it. Then they prepare a document called BRD (Business Requirement Document) which is an exit criterion for this phase. It is a report that details the requirements for a new project. It describes a business need, what is expected as the project proceeds.

1. **Designing:**

Senior developers are responsible in this phase. In this phase the requirements are transformed into a system design. Solution architects and designers design an overview of the software. All technical infrastructure gets finalized in this stage. (WHAT SHOULD BE THE DOMAIN, WHAT SERVER TO USE) The exit criteria for this phase are to circulate a document called the SRS (SOFTWARE REQUIREMENT SPECIFICATION). This document specifies what the software will do and how it is expected to perform.

1. **Implementation and Coding:**

In this phase programmers will write the code based on the design specifications. This phase involves the creation of modules(codes). The exit criteria for this phase are that they should produce the result of unit testing.

1. **Testing:**

In this phase, the software goes through rigorous testing to identify defects and to ensure it meets the specified requirements and performs as expected. The exit criteria for this phase are evidences of completion. For example, a screen shot.

1. **Deployment:**

Once the software has successfully passed the testing phase, it is deployed to the production environment which is made available to the users. The developer is responsible for this deployment. This phase involves installing software, configuring the necessary hardware and software infrastructure and preparing the system for use.

1. **Maintenance:**

After deployment, software will enter the maintenance phase. This involves addressing defects , applying updates and enhancing features. Monitoring of the software happens here. If there are any issues during the monitoring activities.

**STLC (SOFTWARE TESTING LIFE CYCLE) subset of your testing phase of phase 4 in SDLC.**

It is a systematic approach to testing software applications. The goal here is to ensure that the software being developed meets the required standards. STLC involves various phases

1. **Requirement Analysis:**

This phase involves understanding the requirement of the software and identifying testable features of the software application. Write down questions and queries of the software.

1. **Test Planning:**

This phase involves creating detailed test plans that outlines testing approach, objectives, what resources are required and the test deliverables.. It also documents what test levels and the entry and exit criteria for each level of testing. This activity is performed my test managers.

Exit criteria: planning has to be documented and delivered to all stakeholders

TEST PLANNING ACTIVITY

* Analyze the product (examining product features, costs, availability, quality, appearances)
* Design test strategy(which types of testing are to be performed and which entry-exit criteria apply)
* Define test objectives(provides a prioritized list of verification or validation. It is to measure testing progress and verify that testing activity is consistent with project objectives)
* Define test criteria(conditions that a software product muct meet to be accepted by a user, customer or other systems)
* Resource planning(this helps determine the number of resources such as human effort, equipments)
* Plan test environment(once software tests are designed, they need an interface to be executed in. this interface is called the test environment. It is created by integrating hardware, software, proper networking configurations and necessary data to run tests)
* Schedule and estimation(based on the number of resources he will create a schedule of his activies. When will the testing start, provide dates for different testing. Week wise schedule, daily schedule). Number of hours required to work on that functionality.
* Determine test deliverables. He will decide what different documents will be delivered.

TEST DELIVARABLES

* Test plan
* Test case
* Test scenarios
* Test phase

1. **Test Case Development:**

Here test cases are created based on the software requirements. Test cases are designed to validate the functionality, usability, and performance of the software once it is available in the environment.

Test case document is the exit criteria .

**THESE 3 PHASES CAN GO SIMULTANEOUSLY AFTER IMPLEMENTATION OF SDLC.**

1. **Test Environment Setup:**

The testing team sets up the required hardware and software,infrastructure needed to execute the test cases effectively. Confirmation to start the test activity.

1. **Test Execution:**

This is the phase where the actual testing takes place. The testers execute the prepared test cases and record the outcomes and compare them with expected results. Bugs or any other issues encountered during testing are reported to the development team. Reconfirm to the client about the completion of the software.

1. **Test Case Closure:**

In this phase, testers assess whether the objectives have been met. They will determine if further testing is required. If testing is successful, then they close the test case with a “Pass” remark.

**TYPES OF TESTING**

**FUNCTIONAL TESTING:**

It is a type of testing that focuses on verifying whether a system or application performs its intended functions correctly. This involves testing various functions and features of the software to ensure that they work as expected and meet the specified requirements. This type of testing is usually performed after unit testing and integration testing, as it tests the entire system as a whole.

Functional testing can be performed manually or using automated testing tools.

PRIME OBJECTIVE OF FUNCTIONAL TESTING:

* Mainline Function:

Testing the main functions of an application. For example, the log in button. Once you click on the log in button it is expected to show the username and password bar where the user can type.

* Basic Usability:

This involves basic usability testing, which means it check whether a user can freely navigate through screens without any difficulties.

* Accessibility:

This involves testing that checks how accessible is the software for the users.

* Error Condition:

This involves testing to check for error conditions. It checks whether suitable error messages are displayed. For example. If you want to check if error messages occur during typing a wrong password, the tester types in the wrong password to see if error messages occur. This is also called negative testing.

**NON-FUNCTIONAL TETSING:**

It is a type of testing to check non functional aspects such as Performance, usability, reliability, of a software application. For example, non functional testing will check how many people can log in to the system at a given time.

PRIME OBJECTIVE OF NON-FUNCTIONAL TESTING:

* Increase usability, efficiency, maintainability and portability of a product(hidden parameters of a website)
* Reduces production risk and cost associated.
* Non functional testing helps assess how well a system performs under different conditions, such as high loads or stress or whether it can handle expected user volumes.
* It also evaluates the system’s ability to recover from failures, its usability in terms of user interface and user experience.
* This testing leads to a more reliable and robust(strong) software system.

TYPES OF TESTING IN FUNCTIONAL TESTING

* Unit Testing
* Component testing
* Smoke testing
* Sanity testing
* Integration testing
* System testing
* Regression testing
* User Acceptance Testing

**UNIT TESTING**:

This testing is done during the coding phase where the developer write a piece of code and verify whether the code is correct and syncing.

**COMPONENT TESTING** (it is a black box testing technique)

Combination of components become one unit. Could be performed by testers using black box testing. Component testing is done on the section of code. Suppose we have one website, which includes five different web pages. Therefore testing each web page individually is signified as component testing.

**SMOKE TESTING(**build verification testing and Confidence testing)

This testing is done at an earlier stage of development. It determines the deployed software build(code that is delivered to the QA environment) is stable or not( the basic functionalities of the application is working or not).

If smoke testing is failed than we have the right to reject and cannot go any further with other testing.

When new functionalities are developed and codes are deployed we have to perform smoke testing. The aim of smoke testing is to detect early major issues.

If we do not perform smoke testing in early stages, defects may be encountered in later stage where it can be costly.

QA tester does the smoke testing.

Advantages of smoke testing:

* Defects will be identifies earlier
* Improves the quality of the system

**INTEGRATION TESTING**

It is the connection between multiple blocks terms as “ I and T”, “string testing” and “thread testing”

Multiple modules are being developed. Every module is developed by different developers. QA tester tests if these modules are integrated properly.

For example:

Log in module

Username and password module. QA tester will test if log in module is connected with username and password module.

Another example:

Amazon.com

We add products to the cart and when we go to the cart we need to make sure that the selected product is correctly added to the shopping cart.

The integration testing will test to make sure that the selected products are correctly added to the shopping cart.

**SYSTEM TESTING(**END TO END TESTING)

It validates fully integrated software product. It is end to end testing. Interfaced with other softwares and hardwares (connection between different software’s and hardware)

For example: atm machine

It is interfaced with different softwares. There are different softwares and hardware involved here.

Atm machine, host of atm machine(boa), Deutch bank (account holders bank)

Another example:

Transfer money in the account

Sms (receiving msgs

Email (receiving email)

System testing can be tested in a black box level and white box level.

What do we verify?

Test fully integrated application, including external hardware. For eg, gaming console” involvement of headphones and sound. You have to test the game with headphones to see if sound is working.

We also do User experience testing in system testing.

**SANITY TESTING** (BUILD VERIFICATION AND CONFIDANCE TESTING)

This testing is performed when software build is received with minor changes. This is an entry level testing.

Example:

Log in page

Username and password

A code change takes place that introduces OTP(ONE TIME PASSWORD)

In this case system testing will only test the newly added feature i.e the OTP. It wont test the old features.

If sanity test fails, we should not initiate any other testing.

**REGRESSION TESTING(done after functional testing)**

Type of software testing to confirm that a recent added code has not changed the existing feature.

This testing is done to ensure that code changes do not have side effects on the existing functionalities

**When can we use regression testing:**

* When new functionalities are added to the application
* In case of change requirement: when any significant change occurs in the system, regression testing is used. This test is done to check if these shifts have affected other features.
* After a defect is fixed: the tester performs regression testing once the defects are fixed to check whether the changes made while fixing has not affected already existing features
* Once the performance issue is fixed: so once any performance issues after doing non functional testing is fixed, regression testing is done to ensure it had no side effects to other functionalities.
* While integrating with a new external system: regression testing is done whenever the product integrates with a new external system, for example when a POS system is introduced we need to do a regression testing.

**How do we do regression testing**

In order to do a regression testing, we need to first find out defects and identify them. Once the bugs are identified, changes are made to fix the defects and regression testing is done.

* RETEST ALL

This is one of the methods of regression testing. In this test the entire test suite(set) is executed(performed) once again from scratch. This is time consuming and can be very expensive. But this provides a thorough coverage and ensures all functionalities are retested.

* REGRESSION TEST SELECTION

This is a technique where selected test cases from a test case set are executed to test whether the modified code has affected the software application or not.

* PRIORITIZATION OF TEST CASES

In this approach, test cases are prioritized based on whether the impact has a huge difference in the functionality or not. High priority test cases are executed first followed by the low priority.

HOW TO SELECT TEST CASES FOR REGRESSION TETSING

This requires a systematic approach

* Test cases which have frequent defects
* Functionalities which are more visible to the users
* Test cases which verify core features of the product(a feature that is essential to its purpose). It is typically a reason why customers choose and use the product
* Test cases of functionalities which has undergone more and recent changes
* All integration test cases
* All complex test cases (test cases that require higher level of expertise. They typically involve scenarios that are more challenging.

For example, in amazon. Com you add a product then you go the cart. You remove the product. Then add a new product and go back to the cart

* Boundary value test case

Testing the boundaries or limits of input values in order to find out issues relayed to those boundaries.

For example a printer has to make and deliver printed copies ranging from 1 to 150. So to apply boundary value testing, the analysis is done on the boundaries taking the extreme ends where the maximum value is 150 and minimum value is 1. The invalid values in this test case will be 0 and 151.

* A sample of successful test cases( it is accurate and have a clear objective, clearly defined pass/fail remarks)
* A sample of failure test cases(sometimes you have a failed test case, for example, there could be a false positive or a false negative results, or there could be errors due to the environment or a set up issue)

**UAT(USER ACCEPTANCE TESTING)**

Testing technique where users give their acceptance to the software that the IT team has developed. It typically involves testing various scenarios to ensure that the website meets the requirements and expectations of its intended users.

The main purpose of UAT is to validate end to end business flow. It doesn’t focus on cosmetic error, spelling mistakes or system testing. UAT is carried out in a separate testing environment with production- like data setup. It is a kind of a Blackbox testing where two or more users will be involved.

For eg🡪 the log in functionality for an e commerce site;

Validate that the registered users can log in using their credentials. Verify that log in fails for incorrect username and password.

UAT IS PERFORMED BY:

* CLIENT: group who give requirements to the developer
* END USERS

NEED OF UAT

Need of UAT arises once software has undergone unit,integration and system testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them so for testing whether final product is accepted by client/end users, user acceptance testing is needed

* Developers have included features on their own underdstanding
* Requirements changes not communicated effectively to the developers.

Developers code software based on requirement which is their own understanding of the requirements may not actually be what the clients needs form the software.

Requirement changes during the course of the project may not be communicated effectively to the developers

**NON FUNCTIONAL TESTING**

Checks all aspects not covered in a functional testing. Includes performance testing, usability testing, scalability testing and reliability testing.

Non functional test will add value to the product. Demands much more creativity from a tester.

Along with BRD we have to be creative with writing our test cases.

Non functional testing is often automated. It requires special tools. Feedback of customers play an important role.

It checks behavior patterns(focus) of the users applicable for the software.

For example:

While launching a game you need to make sure the age group is appropriate. Age 3-5. The game should be suitable for the mindset of the kids from age 3 to 5. Color combination, how easy it is to access, innovative animation, attractive animation where kids will be focused. Give sporty look

Category based on their business.

* Software for farmers(tracking weather) application suitable for farmers.

Depending on the target customers, we should do non functional testing.

**NON FUNCTIONAL TETSING TYPES (BLACK BOX TESTING AND IS A PART OF STLC)**

1. **SECURITY TESTING: (there is some kind of a code that is needed to do security testing) it is done in a white box testing environment.**

This testing ensures that highly private data/information is not leaked and is secured so that hackers cant get hold of important data. Makes sure that security data is not compromised.

**WHY IS SECURITY TESTING IMPORTANT?**

Since hackers are very common these days, so to avoid hackers attacking the data and important information, the application should be secured to secure customers data.

1. **PERFORMANCE TESTING:**

This type of testing is used to test the speed, response time, stability, reliability(sharing personal data) and scalability(huge number of users) of the application. For eg🡪 how many secs or minutes does a page take to - load from one page to another. Or how many users can log in at the same time before the system crashes. And how reliable is the application for the users to use for a banking application where the users have all their personal data involved.

1. **USABILITY TESTING:**

Method for measuring how easy and user friendly a software application is. Need to make sure that application is targeted to the right group. For example, learning abc’s. this application is targeted for age 1-3.

Another example, gaming application (for age 5-10)

1. **INTEROPERATABILITY TESTING (MULTIPLE SYSTMES COMPATIBLE WITH EACH OTHER)**

This checks that software can interact with other software components. For example, camera on the phone interacts with other softwares such as apps like viber, whatsapp etc.

**DIFFERENCE BETWEEN FUNCTIONAL AND NON FUNCTIONAL TESTING:**

* **FUNCTONAL:** analyzes the functionality, whereas non functional testing analyzes the performance, scalability, reliability and usability aspects of the application
* functional testing is executed in the earlier stages, whereas non functional testing is performed after functional testing.
* functional testing is performed both manually and by using automation tools, whereas non functional testing requires automation tools for effective testing.
* Functional testing focuses on user requirements (BRD), whereas non functional testing focuses on user expectations.
* Functional testing determines what a product is capable of doing( for eg: UBER, how capable it is to book a car), whereas in non functional testing it determines how effectively the product works ( for e.g. UBER, how effectively the application can book a cab)
* Business requirements are the inputs of functional testing, whereas parameters like speed, scalability are the inputs of a non functional testing

Examples of functional testing include unit testing, white box testing, smoke testing, sanity testing, system testing, UAT

Examples for non functional testing include performance testing, stress testing, security testing

**WHAT IS A DEFECT?**

The difference between the expected behavior and an actual behavior of the system is called a defect. Defect is also called an error, fault, problem, an issue and bug.

**TYPES OF DEFECTS**

There are 2 types of defects which is a functional defect and a non functional defect.

1. **FUNCTIONAL DEFECT:**

This is related to a functionality of a piece of software.

* WRONG FUNCTIONALITY

When a functionality is not suppose to be performing in a certain way. For example a login button doesn’t allow users to log in. this is a major functional defect.

* MISSING FUNCTIONALITY

When a function is missing in a very crucial aspect of the application. For example if you are logging in a bank of America account. There is a username, password but captcha is missing.

* UNWANTED OR EXTRA FUNCTIONALITY

An extra functionality or unwanted functionality that is not required by the user. For example, the user requirement has the username, password, and captcha. But there is an extra or unwanted functionality now required by the user which is a fingerprint.

1. **NON-FUNCTIONAL DEFECT:**

* BACKGROUNG COLOR

If text in the application is not readable due to color of the page.

* FONT AND SIZE

If the font used by the developers in not readable or size of the test is too large or too small to read.

* HARD TO USE

For example, if the username is in one side of the page and the password is on the other side of the page. This will be difficult for the users to use the application.

* LOOK AND FEEL

You need to make sure at the aesthetic of the application is present. If its an very untidy and the appearance of the application is messy and there is no alignment.

Severity and priority

[https://www.softwaretestinghelp.com/how-to-set- -priority-and-severity-with-defect-triage-process/](https://www.softwaretestinghelp.com/how-to-set-defect-priority-and-severity-with-defect-triage-process/)

**PRIORITY AND SEVERITY OF THE DEFECT**

There are 2 parameters of effective reporting

* Defect priority
* Defect severity

1. **PRIORITY:**

This type of defect is used in the comparison of two things. One has to be given more importance than others. It has to be resolved first before proceeding to the next one. This type of defect has to be addressed first and it indicates the urgency with which it would need to be fixed. For eg🡪 banking application inaccurately calculates the rates of loan. This can result in customers being charged a higher interest than they should be paying. This could lead to financial losses for the customers.

1. **SEVERITY:**

This type of defect describes the gravity or the depth of an undesirable occurrence.

For eg🡪 severity defect in a banking application could be a security vulnerability that allows unauthorized access to customer accounts. This poses a severe threat to the confidentiality of the data.

**WHO DECIDES THESE PARAMETERS OF PRIORITY AND SEVERITY?**

QA testers decides the severity of the defect and the Business Stakeholders, like the project managers, product managers, BA decides the priority of the defect.

SEVERITY: (TEST ENGINEERS)

* CIRITCAL
* MAJOR
* MINOR
* LOW

PRIORITY (PRODUCT MANAGERS/BA/PROJECT MANAGERS)

* IMMEDIATE
* HIGH
* MEDIUM
* LOW

**DIFFERENCE BETWEEN SEVERITY AND PRIORITY:**

|  |  |
| --- | --- |
| **PRIORITY** | **SEVERITY** |
| Associated with scheduling(time ,schedule) | associated with standards(gravity of the software) |
| It deserves prior attention/order of urgency | it is related to the quality or standards of the coding |
|  |  |

* DUE TO THE SEVERITY OF THE PROJECT, WE WILL NEED TO CHANGE THE PRIORITY OF THE PROJECT.

**LEVELS OF PRIORITY:**

**P1 (IMMEDIATE), P2 (HIGH), P3 (MEDIUM), P4 (LOW)**

**P1:** if the defect is at this level then it has to be fixed within 24 hours. This defect is at P1 level when the defect is found and the feature is completely unusable.

For eg 🡪 in a banking website if the log in button isn’t working then the defect will be categorized as P1, because if the user cannot log in then he wont beable to make transactions, or deposit etc.

**P2:** defects after P1 is categorized as P2. In this level, when a feature is not usable as its suppose to be, due to the program defect.

For eg🡪 amazon. Com, the log in feature is usable, but not in a proper manner. When we log in with the correct password, it is letting us log in, but even if we type the wrong password, it is still letting us log in. this phase can be categorized as P2.

**P3:** this phase involves cosmetic errors.

For eg🡪 in a log in page, if we type the wrong password, it wont let us log in, but the error msg will show “ your password is blank”. So in this phase the functionality is correct but the error msg we get is incorrect. This has to be fixed after P2.

**P4:** defect with low priority in is P4. In this phase there is definitely an issue, but it doesn’t have to be fixed to match the exit criteria. Test closure does not depend on this defect. However P4 must be fixed before Going Live.

For eg🡪 in the log in page, we put in wrong password, it does not log in, there will be an error msg that says password is incorrect. This error msg should be shown in red but the defect here is that the error msg is shown in black. So this is a minor defect that can be fixed after test closure but needs to be fixed before going live.

**LEVELS OF SEVERITY**

**S1(CRITICAL), S2(MAJOR), S3(MINOR), S4(LOW)**

**S1:** defects that completely hamper the feature is placed in s1.

For eg🡪 in the log in page, we enter username and password but the click is not working when we try to log in. this is a major issue in the software that can hamper the user.

**S2:** features implemented that is not meeting its requirements and behaves differently than expected. For eg🡪 in your gmail , you cannot add more than 1 recipient in your CC. this is a s2. Because functionality is there but cannot add more than 1 recipient.

**S3:** feature implemented that is not meeting its requirement and behave differently than expected, but the impact is negotiable to some extent.

For eg🡪 in gmail application, in terms and conditions, there are many links available , but if one of the links in terms and conditions is not working then this defect will fall in this category.

**S4:** cosmetic defect including spelling mistake, the font, color, alignment. In this phase the feature is not impacting the behavioral or the functionality of the application.

For eg🡪 in gmail application, the licence page may have some spelling mistakes or mis alignment in the page. This defect is classified as low.

**EXAMPLE OF S4-P1(LOW SEVERITY AND HIGH PRIORITY)**

In any ecommerce site, if their logo is wrong or there is a spelling mistake then this is categorized as P1 because this is going to affect their brand. but since it’s a cosmetic error and no features in the application is getting affected, this category will be put in S4.

**EXAMPLE OF S1-P3(HIGH SEVERITY AND LOW PRIORITY)**

For example, if you go to the amazon website and you are trying to go to their career page to look for vacancies and hiring department, it does not let you log in and gives you an error when logging this means that this function is not working.

In this case the severity becomes critical but the priority becomes P3 because the main purpose of the website is to sell products.

**EXAMPLE OF S4-P4(L LOW SEVERITY AND LOW PRIORITY)**

In a an ecommerce website there could be a minor issue in the email notifications sent to customer after making a purchase.

This can be considered a low severity because it does not affect the core functionality of the ecommerce platform.

This can also be considered a low priority because it does not have an immediate impact on the operation of the ecommerce business or customer experience.

**EXAMPLE OF S1-P1( HIGH SEVERITY AND HIGH PRIORITY)**

In an ecommerce website, there could be a situation where the customers are unable to complete the check out process due to a critical error in the payment gateway process. This defect is considered high severity because it prevents customers successfully finalizing their purchase , which directly impacts the core functionality of the ecommerce website and can potentially be left with unsatisfied customers.

This defect can also be assigned a high priority because this will cause a huge revenue loss and can be left with unsatisfied customers and could also damage the reputation of the business.

There are four possibilities:

S1/p1

P1/s3

S1/P4

S4/P4

**EXAMP**

**DEFECT LIFE CYCLE**

Bug life cycle

<https://www.softwaretestingmaterial.com/bug-life-cycle/>

also known as the bug life cycle. The bug life cycle consists of various stages that a defect goes through. The bug cycle should go through a life cycle to be closed.

1. **NEW**

This is an initial stage when a bug is identified then the status become “NEW”.

1. **ASSIGN**

In this stage the bug is assigned to a developer to resolve to address and resolve it. The status then changes to ASSIGN

1. **OPEN**

When the bug is actively being worked on by the developer. In this stage they analyze the bug and debug the code and make necessary changes. The status changes to OPEN.

1. **FIXED**

When the developer makes the necessary code changes and verifies the changes, the status changes to FIXED. After this the bug is passed to the testing team.

1. **TEST**

As soon as the bug is passed to the testing team, the team will re-execute the test to ensure the defect has been fixed. During this activity the status changes to TEST.

1. **VERIFIED**

Once the issue is fixed by the developer and retested by the tester and he found that the bug has been fixed in an expected manner the status becomes VERIFIED

1. **CLOSED**

Once status is verified and bugs no longer exist the status becomes CLOSED

1. **REOPEN**

If the tester found the defect is not completely fixed and the bug still need to be fixed partially or fully then the status changes to REOPEN. The cycle will again go through OPEN, FIXED, VERIFIED AND CLOSED.

1. **DUPLICATE**

During the open phase, after analyzing the defects, if the developer found that this defect was already raised by the QA team then the status would go to DUPLICATE. Then the case will get CLOSED.

1. **DEFERRED**

At the time of analyzing the defect, if the bug is not related to the current build or the bug is not important to fix it immediately or if the customer is planning to change the requirements, or if the bug is expected to be fixed in the next release, the project manager or the lead will mark the status as DEFERRED.

1. **REJECTED**

If the system is working as expected and the bug is just due to some misinterpretation such as an extra feature or a old requirement then the developer can mark the bug as REJECTED AND CLOSE the case.

THERE ARE ALSO OTHER HIDDEN STATUSES SUCH AS:

1. **CANNOT BE FIXED**

Due to technology limitations, or like any production issue or if it tends to get really expensive than the status changes to CANNOT BE FIXED

1. **NON REPRODUCABLE**

Due to platform mismatch, or a build mismatch or even if there is an inconsistent defect then the status goes to NON REPRODUCABLE

1. **NEED MORE INFORMATION**

If the developer is unable to reproduce a bug as provided by the tester, then the developer can change the status to NEED MORE INFORMATION. In such cases, the tester will need to ass more detailed steps and then assign the bugs back to the development team to fix it.

**PRINCIPLES OF SOFTWARE TESTING**

It is very important to achieve a test results while conducting software testing. So there are 7 principles of software testing.

1. **TESTING SHOWS THE PRESENCE OF DEFECTS**

If you perform testing then you can identify any errors or bugs. Each and every test cases should be traceable. Testing should be linked with the requirement. By doing testing, we can decrease the number of bugs. This does not mean that the application is bug free. But we can take precautions and make the software product 99% bug free.

1. **EXHAUSTIVE TESTING IS IMPOSSIBLE**

Sometimes it is very difficult or next to impossible to test all the positive and negative combinations of the data. For eg🡪 if you are working on a banking application and test the feature to transfer money from one account to another, lets say around $10,000. So while testing this feature, you cannot test putting every number to test whether the feature is working or not.

1. **EARLY TESTING**

As a tester, if we start identifying the bugs in the initial phase, that is the requirement phase. We should review the requirement document and if there is any issue that arises, then we have to fix it at that stage, so that the developer will not develop the wrong code and it will tend to get more expensive if we identify defects in the future phase.

1. **DEFECT CLUSTERING**

When a defect is concentrated in a single module, it is called a defect clustering. This usually happens in a complex feature of the application. For eg🡪 payment feature can have more defect clustered, because it is a complex feature. Like small defects like not taking the card number might eventually affect or lead to defects in other features of the application.

1. **PESTICIDE PARADOX**

This name comes from the pesticides that farmers use. Using pesticides overtime will lead to the insects developing a resistance to the pesticide. In the same way, in the software testing world, if the same set of test cases are being conducted repeatedly, the method will not work and it will be useless for discovering new defects. In order to eradicate this, we need to implement new test cases which will help find new errors or bugs.

1. **TESTING IS CONTEXT DEPENDENT**

There are different kinds of business fields, like an ecommerce application, banking application, gaming application, commercial or tourism. Every business application will have their own features and functionalities. And there are different kinds of testing available for every business application.

For eg🡪 gaming application and a banking application will have a completely different kinds of testing available. Like, for gaming application, test engineers should understand the features and expectations of the users. The age group for the gaming application.

And for banking application, the test engineers should have a different kinds of test to ensure that important data of customers are secured.

1. **ABSENCE OF ERRORS FALLACY**

When a tester performs a series of test and has not encountered any errors or bugs, and based on this assumption the tester concludes the application to be bug free and is ready for release without further testing. The tester ignores the possibilities that there still might be undiscovered bugs that have not been encountered yet.

The absence of errors does not guarantee a 100% bug free application. To avoid this, the tester should do a comprehensive testing, such as code reviews and also perform automated testing to identify potential errors.

[**https://www.browserstack.com/guide/verification-and-validation-in-testing**](https://www.browserstack.com/guide/verification-and-validation-in-testing)

**VERIFICATION**

Verification refers to the process of evaluating whether the system meets the specific requirements. It is responsible for the accuracy of the software. It involves reviews, inspections and meetings. This is carried out by the QA team. The execution of the code is not performed here and verification is carried out before the validation process.

WHEN TO USE VERIFICATION TESTING

This must be run after every feature is implemented. For eg🡪 a button about us page, before creating this page, verification tests will review the requirements of this feature and some brainstorming phases

**VALIDATION**

Validation is the process of evaluating the software at the end of the development process whether software meets the customer expectations or not.

This involves, black box testing, white box testing and grey box testing. The validation process is carried out by the testing team. Code is executed and is carried out after the verification process.

WHEN TO USE VALIDATION

Validation tests must run after every feature in the development process is completed.

Unit testing and integration tests that are a form of validation tests is run after every code has been created. Testers also check whether all the functions and features as expected on different browsers. For example. The about us page, is it working perfectly in google chrome, safari or iphone, android.

**ADVANTAGES OF VERIFICATION.**

It contributes to the overall quality and reliability of the software.

* Early detection of errors: code reviews and inspection help identify errors in early phase. It becomes easy and more cost effective.
* Verification ensures that software meets the specified requirements, by thoroughly checking the software against the requirement documents.
* Verification activities including testing help discover and fix defects that lead to overall improvement in the quality of the software.

**ADVANTAGES OF VALIDATION**

* Any bugs missed during verification will be detected in validation.
* If specifications were incorrect then validation will reveal their inefficiency.
* Validation tests ensures that the product matches to customer demands
* This test also ensures the software functions flawlessly across different browsers, devices.

**TEST SCENARIOS AND TEST CASES**

<https://www.javatpoint.com/test-scenario>

Test Scenario

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

**TEST SCENARIO FORMAT:**

SERIAL NO

SCENARIO ID

MODULE NAME(

TEST SCENARIO (THE ACTION A USER MAY UNDERTAKE IN THE APPLICATION)

NUMBER OF TEST CASES

PASSED TEST CASES

FAILED TEST CASES

PRIORITY

COMMENTS

**HOW TO WRITE A TEST SCENARIO**

* Read the requirement doc (BRD, SRS,FRS) of the software.
* Determine all technical aspects and objectives for each requirement.
* Find all possible ways by which the user can operate software.
* Determine all possible scenarios due to which system can be missed and also detect the users who can be hackers (Identify possible scenarios that can be misused)
* After reading the required doc and completion of required gathering activity make a list of different scenarios to verify each function of software.
* Supervisor of the project reviews all scenarios. Later they are evaluated by other stakeholders of the project.

**FEATURES:**

* One line scenario that guides you about the sequence.
* It reduces complexity and repetition of the product.
* It is a time saver activity.

**RULES WHEN WRIITNG A TEST SCENARIO:**

* Always list down the most common used feature and module by the users ( start with high priority)
* Start writing the test scenario module by module, so we don’t miss out anything. For eg🡪 write log in module first, then inbox module, then compose.
* Scenarios are module level.
* Delete scenarios should be the last option to write, else we will waste lost of time in creating the data once again.( first write the upload module then the delete module in any functionality)
* Should be written in simple language.
* Scenario should be written in one line or maximum of two lines and not in paragraphs.
* Scenarios should consist of Do’s and checks(expectations)

**TEST CASE**

[**https://www.javatpoint.com/test-case**](https://www.javatpoint.com/test-case)

[**https://www.guru99.com/test-case.html**](https://www.guru99.com/test-case.html)

Test case is a subset of test scenario.

Test case is how you are going to test the application. It is a detailed document that contains all possible inputs(positive as well as negative), which are used for test execution process.

Test case gives a detailed information about testing strategy, testing process, preconditions and expected output. We should have a test case template to maintain the uniformity to prepare test documents.

The template includes **TEST CASE ID, MODULE NAME (LOG IN), TEST SCENARIO, REQUIREMENT #, TEST CASE DESCRIPTION (THE ACTIONS REQUIRED TO VERIFY A SPECIFIC FEATURE OR FUNCTIONALITY IN SOFTWARE TESTING), PRECONDITION(CRITERIA THAT MUST BE MET BEFORE A PARTICULAR TEST CASE), STEPS, TEST DATA(SPECIFIC INPUTS THAT ARE USED TO EXECUTE A TEST CASE, FOR EXAMPLE USERNAME OR PASSWORD), EXPECTED RESULT, POST CONDTION, ACTUAL RESULT, TYPE(TYPE OF TESTING),PRIORITY(HIGH OR LOW), COMMENTS, AUTHOR, REVIEWED BY AND EXECUTOR.**

**WHEN DO WE WRITE A TEST CASE?**

When the customers gives the requirement document to the developer and in the meantime the testing team will start writing the test case with the help of the BRD.

**WHY DO WE WRITE A TEST CASE?**

* Require consistency of test execution (quality)
* To make sure a better test coverage(the depth at which we are testing)
* Depends on the process rather than one person(every person has their own maturity level of understanding the requirement which ends up with different depths of understanding the requirement document.)
* To avoid training for every new test engineer on the product(if test cases are documented than new employees can understand from the test case document)

**BEST PRACTICE FOR WRITING GOOD TEST CASE(FEATURES)**

1. **Should be simple and transparent:**
2. **While writing the test case we need to keep the end user in mind:** the ultimate goal is to create an application that will be easy to use and operate for users, so that’s why test case should be written keeping in mind the perspective of the user.
3. **Avoid test case repetition:**
4. **Never assume**: a tester should never assume functionalities and features of the application. That is why BRD is provided**.**
5. **Ensure that there is 100% coverage**: means use traceability matrix to ensure that no conditions are left untested.
6. **Test case should be numbered:** test cases should include test case ID so it can be easily identified while tracking defects.

**REQUIREMENT TRACEABILITY MATRIX (RTM)**

This is a high level document that helps to map and trace user requirement with the test cases and to ensure that each and every requirement is tested thoroughly.

**WHY DO WE PREPARE RTM/ WHY DO WE NEED RTM/IMPORTANCE OF RTM**

* Link test cases, requirements and defects accurately
* Provide good coverage data
* Assures good quality of the application to the customer

**TYPES OF RTM**

**Forward traceability :** this is most commonly used. This ensures that requirements are mapped to the test cases. So rows will include requirements and columns will include test cases

**Backward traceability:** this ensures that test cases are mapped to the requirements. So rows will include test cases and columns will include requirements.

**Bi-directional traceability:** this includes both forward and backward traceability. This ensures that there us a clear and complete understanding of how requirements are covered by test cases and how test cases are linked back to the requirements

**IMPOTANT PARAMETERS OF RTM**

**RTM** typically contains important parameters that link requirements and other features

* **Traceability # (serial #) …………..IMP**
* **Requirement ID from BRD…………..IMP**
* **Requirement description: a clear description of the requirement**
* **Technical requirement ID form SRS…………….IMP**
* **Test case ID……………IMP**
* **Test design (whether test case is written or not written)**
* **Designer (person who has written the test case)**
* **UAT execution(software is tested in the real world, by intended users)**
* **Product execution**
* **Defect (yes or no)**
* **Defect ID**
* **Defect status (fixed or open)**
* **Requirement coverage status(whether this requirement is completely covered or partially covered)**

**BOUNDARY VALUE:** <https://www.youtube.com/watch?v=DpDgaGP-jsQ>

It is a software testing technique. It reduces number of test cases and achieves 100% test coverage. Since its not possible to perform exhaustive testing with positive and negative inputs this is where boundary value analysis comes into use. This saves time and resources and makes the test cases more manageable. It is a black box testing and goes hand in hand with equivalence partitioning

GUIDELINES FOR BVA:

While testing with BVA we need to keep in mi nd the formula that should be used in the test

X-1, X, X+1……………………………Y-1, Y, Y+1

Where x is the lower value and y is the higher value.

Suppose a printer has to make printed copies ranging from 1 to 50,so to apply boundary value testing, the analysis is done on the boundaries taking the two extreme ends. The maximum value is 50 and the minimum value is 1

So when we apply to the formula the value of x is 1 and the value of y is 50

1-1, 1, 1+1…………………50-1, 50, 50+1

0,1,2……………49,50,51

So here there are 4 test data and the invalid values in this test case will be 0 and 51.

**EQUIVALENCE PARTIONING.**

This is also a software testing technique. and a black box testing technique. this can be applied to all levels of testing like unit, integration and system testing. In this technique input data is divided into equivalent partitions that can be used to derive test cases.

Lets take an example of an ecommerce website. the portal has a discount of 5%, 10% ,20% and 25%. It says that the customer spending from $1-$100 will get a 5% discount, customers spending $101-$200 will get a 10% discount and customers spending from$201-$400 will get a 20% discount. And then anything above %401 up to $1000 will get 25% discount

So now this input will be divided into equal partition. So the 5%,10%,20% and 25% are all valid partitions. But what about customers that have purchase less than $1 say .99 then that would be considered an invalid partition because the purchase is not allowed. And because the max purchase a customer can do is $1000, then customers who purchase products for more than $1000 will not be allowed any discount and would be considered as an in valid partition. So now what we as testers can do is that take random numbers from each partition and test the values. This gives us a good coverage testing.

**DIFFERENCE BETWEEN BOUNDARY VALUE ANALYSIS AND EQUIVALENCE PARTITIONING**

**Boundary value analysis** emphasizes testing the boundaries and limits of input values. This aims to identify defects that occur at the boundaries of accepted ranges

**Equivalence partitioning** focuses on dividing inputs into equivalent parts that group similar inputs together and assume that an input within the particular partition detects an error.

**WHAT IS BRD/BRS**

BRD stands for Business Requirements Document. It is a high level document of a project. It serves as a communication tool between stakeholders and clients to ensure better understanding of the objectives of a project.

**BRD TYPICALLY INCLUDES (IMPORTANT CHAPTERS/SEGMENTS OF BRD)**

**Introduction :** this provides an overview and purpose of the document

**Scope:** this explains the boundaries of the project. What is included int eh project and what is excluded.

**Functional requirements:**  describes the specific functionalities that the project must deliver

**Non functional requirements:** this specifies the characteristics of the system such as performance, security, usability, scalability and reliability.

**Constraints:** this identifies any limitations that can impact the project, such as budget, technical issues, etc.

**Assumptions:** this states any assumptions made during the requirement gathering process.

**Risks:** this identifies certain risks or challenges associated with the project for eg, overhead expenses, bargaining power,

**Business requirements:** the requirements in the document will be prioritized as critical, high, medium and low

**Stakeholder sign off:** space for stakeholders to review and approve the document.

**WHAT IS SRS**

SRS stands for Software Requirement Specification. This document describes what the software will do and how it will be expected to perform.

**SRS TYPICALLY INCLUDES (IMPORTANT SEGMENTS/CHAPTERS OF SRS)**

**Introduction:** this provides an overview of the software system. Includes background information of the software.

**System overview:** this describes the system’s high level components and interaction with external systems.

**Functional requirements:** this specifies specific functions, features and capabilities of the software.

**Non functional requirements:** this describes the characteristics of the software system such as performance, scalability, usability, reliability, security and compatibility.

**User interface:** includes diagrams, screen layouts and user interaction guidelines.

**Testing and Validation:** this includes test scenarios, test cases and acceptance criteria.

**DECISION TABLE TECHNIQUE** (javatpoint)

<https://www.youtube.com/watch?v=RrJXLdv1i74>

Decision table technique is one of the widely used case design techniques for black box testing. This is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form.

This technique is used to pick the test cases in a systematic manner; it saves the testing time and gives good coverage to the testing area of the software application.

**WHY IS DECISION TABLE TECHNIQUE IMPORTANT? WHEN DO WE USE DECISION TABLE TECHNIQUE?**

 It helps to test different combinations of conditions and provides better test coverage for complex business logic. When testing the behavior of a large set of inputs where system behavior differs with each set of inputs, decision table testing provides good coverage and the representation is simple so it is easy to interpret and use.

In Software Engineering, boundary value and equivalent partition are other similar techniques used to ensure better coverage. They are used if the system shows the **same**behavior for a large set of inputs. However, in a system where for each set of input values the system behavior is **different**, boundary value and equivalent partitioning technique are not effective in ensuring good test coverage.

This is where decision table technique comes to use. This technique can make sure of good coverage and is simple to use and interpret.

For example: A log in screen with email and password input boxes. There are two conditions here, one is the email and the second is the password. The condition is simple, the user will be routed to homepage if they give the right username and password. But a warning will appear if any of the inputs are incorrect.

|  |  |
| --- | --- |
| email | password |
| t | T == T |
| f | T == F |
| t | F == F |

The decision table provides a structured way to define different conditions and corresponding outcomes based on the given criteria.

**ADVANTAGES OF DECISION TABLE TECHNIQUE**

* When the system behavior is different for different inputs and not the same for a range of inputs, both equivalent partitioning, and boundary value analysis won’t help, but a decision table can be used.
* The representation is simple so that it can be easily interpreted and is used for development and business as well.
* This table will help to make effective combinations and can ensure better coverage for testing
* Any complex business conditions can be easily turned into decision tables
* In a case we are going for 100% coverage typically when the input combinations are low, this technique can ensure the coverage.

**STATE TRANSITION TESTING**

This is a software testing technique that focuses on testing the behavior of a system as it transitions between different states. It focuses on testing the system’s behavior as it moves from one state to another.

Let’s consider an ATM system function where if the user enters the invalid password three times the account will be locked.

In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

There are four different elements included in the test cases while performing state transition testing? WHAT ARE THE FACTORS FOR STATE TRANSITION TESTING?

1. **Initial State:** specify the starting state of the software.
2. **Events:** identify the triggers that cause state transitions. For example, it could be user inputs, system generated events.
3. **Transitions:** expected transition from one state to another. So each transition represents a change in the system.
4. **Actions:** this results from a transition. So the error message you get if you type in an invalid password. These actions could be system operations, outputs or changes in the user interface.

**The goal of state transition testing is to ensure that the system behaves correctly as it transitions between states and handles events appropriately.** By applying this testing technique testers can identify issues such as incorrect state transitions, missing or incorrect actions and unexpected system behavior.

**ERROR GUESSING TECHNIQUE**

This is an informal testing technique where testers solely rely on their past experiences and knowledge to identify potential defects in the system. Testers “guess” where errors might occur based on their understanding of how the system works.

The error guessing technique does not follow a structured test cases. Instead it relies on the testers’ ability to imagine a scenario that might lead to errors.

For example: Input fields often have maximum length restrictions and exceeding those limits may cause errors or unexpected behavior. Registering with a name that exceeds the maximum character limit will you an error message.

**WHAT ARE THE FACTORS IN ERROR GUESSING?**

Some of the key factors to consider in error guessing technique are:

1. Tester experience: a tester with a deep understanding of the system is more likely to identify potential error scenarios accurately.
2. Domain familiarity: if the tester is familiar with the domain, this allows the tester to find potential errors in the system
3. If the testers have knowledge of the past defects related to similar projects this can help testers to focus their error guessing efforts.
4. Complex features are more likeable to have defects, and testers can use this information for error guessing technique

**SOFTWARE TESTING METRICS**

Software testing metrics are indicators that is used to measure progress, quality, productivity and the overall health of software testing. The purpose of this testing technique is to increase the efficiency and effectiveness of the testing process while also providing accurate data about the testing process.

**IMPORTANCE OF METRICS IN SOFTWARE TESTING:**

* Test metrics help to determine what types of improvements are required in order to create a defect free product.
* To make informed judgements about testing phases that follow, such as project schedule and cost estimation.
* Examine the current technology and see if any changes are needed.

**TYPES OF SOFTWARE TESTING METRICS**

**PROCESS METRICS:** this is critical to SDLC process’s improvement and maintenance. Here the characteristics of the projects are defined.

**PRODUCT METRICS:** in this metrics a product’s design, performance, quality and complexity are defined.

**PROJECT METRICS:** project metric is used to evaluate the project’s overall quality. It is used to determine cost, productivity and the flaws of the project.

**IMPORTANT METRICS**

* **Defect Metrics:** this helps the developers to understand the aspect of the software such as functionality, performance, stability, usability, compatibility.
* **Schedule Adherence:** the major purpose of this metrics is to determine the time difference between a schedule’s expected and actual execution time. For eg: if the expected schedule was for 1st jan but the actual execution time was done on 5th of jan. so in this metrics they determine how many days they were behind the scheduled date.
* **Defect Severity:** here it allows the developer to understand how severe the defects are in the software.
* **Test case efficiency:** this is used to measure how effected test cases are at detecting errors.
* **Defects finding rate:** this is measured to check the number of defects found over a period of time. For eg, how many defects are found daily, or weekly.
* **Defect fixing time:** measures to to check the amount of time it takes to fix a bug
* **Test coverage:** this specifies the number of test cases assigned to a program. Also ensures that the testing are completely concluded
* **Defect cause:** this helps to find the reason of the defect in the system

**TEST METRICS LIFE CYCLE**

Test metrics life cycle is written by the managers. There are various stages in the test metrics life cycle.

1. Analysis:

Identication of the metrics. Define the specific metrics you want to track during the testing process, such as test coverage, defect severity etc.

1. Communicate:

This is a one time activity. In this phase you communicate with the stakeholders and inform the need of metrics. Also need to educate the testing team on the data that must be collected to process the metrics.

1. Evaluation:

In this phase, data should be acquired and verified. After acquiring the data that has been collected, we need to calculate the value of the metrics.

1. Report:

This is the last phase of the life cycle. In this phase we need to come up with a conclusion and distribute the report to stakeholders, and gather their input of the value of metrics.

**FORMULA FOR TEST METRICS:**

1. **Percentage test case execution=(no of test cases executed/ total no of test cases written)x100**

**(164/40)x100 = 410**

1. **TEST CASE EFFECTIVENESS**

= **(No of defects detected/no of test cases run) x 100**

(20/164) \* 100

12.19

1. **PASSED TEST CASE PERCENTAGE**

= **(Total no of test cases passed/total no of test cases executed) x 100**

(100/164)\* 100

60.97

1. **FAILED TEST CASES PERCENTAGE**

= **(Total no of failed test cases/total no of tests executed) x 100**

(60/164)\*100

36.58

1. **BLOCKED TEST CASES PERCENTAGES**

= **(Total no of blocked tests/ test no of tests executed) x 100**

(4/164)\*100

2.4

1. **FIXED DEFECTS PERCENTAGES**

= **(Total no of flaws fixed/ no of defects reported) x 100**

(12/120)\*100

10

1. **REWORK EFFORT RATIO**

= **( Actual rework efforts spent in that phase/ total actual efforts spent in that phase) x 100**

(12/164)\*100

7.31

1. **ACCEPTED DEFECTS PERCENTAGE**

= **( Defects accepted as valid by dev team/ total defects reported) x 100**

(15/20)\*100

75

1. **DEFECTS DEFERRED PERCENTAGE**

= **(Defects deferred for future releases/ total defects reported) x 100**

(5/20)\*100

25

**TEST METHODOLOGIES**

**WATERFALL MODEL:**

Waterfall model is an SDLC approach. It is also called the linear sequential model, which means it flows from one phase to another phase. In this model, each phase must be completed before the next phase can begin. Which means that the output of one phase acts as the input for the next phase. There is no overlapping in the phases. This model is used in small projects.

**PHASES OF WATERFALL MODEL**

1. **REQUIREMENTS GATHERING AND ANALYSIS:**

All possible requirements of the system that are to be developed are obtained in this phase and are documented in the requirement specification document. This is a very crucial phase because waterfall method is a sequential linear model, and each phase depends on completion of the previous one. So gathering requirements at the beginning ensures that the subsequent phases can proceed smoothly. Because of this phase, the developers can design, develop and test the product based on the requirements , minimizing delays.

1. **DESIGN:**

After the requirements gathering phase, the system design is prepared. This phase helps in logical design i.e how software data flows and physical design on what hardware to use.

1. **IMPLEMENTATION:**

In this phase, after inputs from the design phase, the actual software is implemented.

1. **VERIFICATION:**

In this phase the software is tested against every requirement.

1. **DEPLOYMENT:**

Once the functional and non functional testing are done, the product is ready to be deployed to the customers or released in the market.

1. **MAINTENANCE:**

During the customer environment, there can be issues, to solve these issues, maintenance is done to deliver these changes in the customer environment.

**ADVANTAGES OF WATERFALL**

* It is simple and easy to understand
* It is a very simple, model and due to this, it is very easy to manage
* Works well for very small projects where requirements are crystal clear
* There are clearly defined stages
* Process and results are well documented.

**DISADVANTAGES OF WATRFALL MODEL (What are the problems faced in the waterfall model?)?**

* The waterfall model follows a linear sequential approach, where each phase must be completed to proceed to the next, because of this rigid structure it is difficult to accommodate any changes during the development process
* Testing is done in the later stages, due to this discovery of issues can cause delays and increase the cost in fixing them.
* In this method, the final product is delivered only after the entire development process is complete and due to this the stakeholders cannot see any progress nor provide any feedbacks during the life cycle
* There is limited customer involvement

**WHEN MIGHT YOU USE THE WATERFALL METHODOLOGY FOR THE SOFTWARE DEVELOPEMNT LIFE CYCLE PROCESS?**

When requirements are well documented, clear and fixed

* When product definition is stable
* When the project is not dynamic where the requirement are not uncertain

**WHY DOES THE REQUIREMNT PHASE IN THE WATERFALL METHODOLOGY MATTER?**

Since waterfall methodology is a linear sequential model, that means it flows from one phase to another phase, the requirements help establish a clear and well defined scope for the project. The requirement phase outlines what features and functionalities the product should include. This sets realistic expectations for both the development team and the client that prevents any problems in the later stage of the project.

**HOW DO YOU DECIDE WHETHER TO CONDUCT A SOFTWARE PROJECT USING AGILE OR WATERFALL METHODOLOGY?**

1. Waterfall: If the project requirements are well defined stable and unlikely to change during the project

Agile: if the requirements are complex, evolving or may change during the development process.

1. Waterfall: if the customer doesn’t need continuous involvement in the development process

Agile: if the customer wants to be active and provide feedback and be part of the regular iterations.

1. Waterfall: if the project is relatively small and has a clear scope

Agile: if the project is large, complex and can be broken down into iterations.

1. Waterfall: if the project has a fixed timeline and deadline is crucial

Agile: if the project timeline is more flexible and can accommodate iterative development

**How do you know when to progress to the next stage of a project when using the waterfall method?**

Waterfall is a linear sequential model. So every phase has a entry and an exit criteria to proceed to the next phase.

For example during the requirement phase, once the BRD document s provided you can proceed to the next phase, which is the design phase which is the exit criteria of the requirement phase. Similarly in the design phase, the exit criteria for this phase is to circulate a document called the SRS (SOFTWARE REQUIREMENT SPECIFICATION) to proceed to the implementation phase .This document specifies what the software will do and how it is expected to perform.

**V- MODEL**

. Also known as the verification and validation model. V model is an extension of the waterfall model and every stage corresponds to the testing phase. This is a highly rigid model and the next phase starts only after the completion of the previous phase.

The V model is planned in parallel. So, the verification phase on one side and validation phase on the other side.

The verification phase includes:

Requirement analysis

System design

Architecture design

Module design

After the module design there is a coding phase. The coding phase goes through numerous code reviews.

Then there is the validation phase that includes:

Unit testing, designed in the module design phase.

Integration testing, this is associated with the architecture design

System testing, this is associated with the system design phase

Acceptance testing, associated with the requirement analysis phase

**ADVANTAGES OF V MODEL(benefits)**

* Like the waterfall model, this is a highly disciplined model where phases are completed one at a time
* Works well for small projects
* Simple and easy to understand and use
* Plans for verification and validation in the early stages of development

**DISADVANTAGES OF V MODEL(WHAT ARE THE PROBLEMS FACED IN THE V MODEL)**

* Not for complex and big projects
* Not suitable for projects where requirements have a high risk of changing
* Once application is in the testing stage, it is difficult to go back and change the functionality
* Working software will only be produced after the completion of the life cycle.
* V model may not provide opportunities for customer feedbacks. Customers do not get a chance to interact with the team, nor see the progress made during the process
* Making changes could be time consuming and costly.

**WHEN MIGHT YOU USE THE V MODEL METHODOLOGY FOR THE SOFTWARE DEVELOPMENT LIFE CYCLE PROCESS?**

* When projects are very small and there are crystal clear requirements
* When customers are sure that no changes will take place during the development process
* When you want to detect defects and issues as early as possible in the development process
* If the project has minimal customer involvement then v model can work well

**WHY DOES THE REQUIREMENT PHASE IN THE V MODEL METHODOLOGY MATTER?**

Just like the waterfall methodology, v model is also a linear sequential model. You can only move on to the next phase once the previous phase is complete.

This means that during the requirement phase, requirements from the customers is the most important aspect of the project. The requirements have to be crystal clear to the stakeholders so no changes have to be made in between the process which might delay the process and can also get expensive.

The requirements are also validated during the phase to ensure that it is consistent and feasible. And every validation helps in identifying potential issues and resolving them before moving forward. The V model emphasizes on traceability, where each requirement is linked to test cases. This ensures that every requirement is thoroughly tested. And lastly when requirements are well understood and met then the likelihood of customer satisfaction increases

**HOW DO YOU KNOW WHEN TO PROGRESS TO THE NEXT STAGE OF A PROJECT WHEN USING THE V MODEL**

Like waterfall methodology, in v model the progression to the next stage is only determined by a successful completion of the previous stage. To know when to proceed to the next stage, an exit criteria must be met.

For example; in the requirement phase, BRD should be an exit criteria to proceed to the next phase which is the system design. And to proceed to the next phase from the design phase, the design document which is the SRS should be completed to proceed to the next phase.

**ITERATIVE MODEL:**

An iterative model is a software development methodology that breaks down the development process into series of cycles or iterations. In this model, you can start with the software development in the first version. After the first version, if any changes needs to be made then a new version is created with a new iteration. Each of these iterations involves a complete development cycle on its own which includes design, development, testing and review. In this model, planning and requirement analysis, deployment and maintenance are not a part of the iterations. These phases will be taken place only once.

The flow of this model will include:

REQUIREMNT 🡪ANALYSIS

DESIGN

TESTING

IMPLEMENTATION

REVIEW

If any changes are made then a second version is created with the new iteration that start with:

DESIGN

TESTING

IMPLEMENTATION

REVIEW

This goes on until the process is finalized, then comes your DEPLOYMENT and then MAINTENANCE

**ADVANTAGES OF ITERATIVE MODEL (BENEFITS)**

* Testing and debugging are done is smaller iterations
* Parallel development can be planned
* Easily acceptable model for frequently changing requirements
* User feedback on each iteration
* Product is built step by step. So defect tracking is possible in early phases

**DISADVANTAGES OF ITERATIVE MODEL(PROBLEMS YOU MIGHT FACE IN THE ITERATIVE MODEL)**

* More resources are required
* It is not suitable for smaller projects
* Requirement changes can cause over budget
* Project completion date cannot be confirmed because of continuous requirement changes

**WHEN MIGHT YOU USE THE ITERATIVE MODEL FOR THE SOFTWARE DEVELOPMENT LIFE CYCLE PROCESS?**

* When the project is big and complex and the requirements are most likely to change
* When the team has less experience in the technology

**WHY DOES REQUIREMENT PHASE IN ITERATIVE MODEL MATTER?**

The requirement phase in the iterative model matter because, the requirement phase sets the initial direction and goals of the project. It provides a foundation upon which the iterations are based. Clear and well defined requirements ensure that each iterations focuses on delivering specific objectives. Engaging stakeholders including customers during the requirement phase fosters collaboration and allows early feedbacks. And properly documenting requirements in the requirement phase sets the stage for verifying that each iteration meets the specified criteria.

**HOW DO YOU KNOW WHEN TO PROGRESS TO THE NEXT STAGE OF A PROJECT WHEN USING THE ITERATIVE MODEL?**

Each iterations has a pre defined goals. So before moving on to the next iteration, the development team needs to make sure that the goals that were set for the current iteration has been met. These goals could include delivering specific features or addressing high priority requirements.

Before advancing to the next stage, it is crucial to collect feedbacks from the stakeholders including the customers and the end users. And validate whether the delivered features met the customers expectations.

In the iterative model, before advancing to the next stage, the software should go through a thorough testing to ensure that it meets the desired level of quality.

**AGILE SCRUM:**

Agile model is a software development approach which is based on iterative development. Agile model breaks into smaller iterations. Each of these iterations is considered as a short time frame which typically last about 1-4 weeks. This helps to minimize the project risk and reduce the overall project delivery time. One iteration involves phases of planning, design, development, test, deployment and then the review. Here review only happens after the deployment.

**ADVANTAGES OF AGILE MODEL: (benefits)**

* There is an increased customer satisfaction, since the feedbacks from the customers are considered after every scrum
* There is a daily interaction between the team and the client
* Change in requirements are welcome in the later stages if required by the customer
* Improves the quality of the product
* Works better in the environment that keeps changing frequently
* There is a faster detection of bugs

**DISADVANTAGES OF AGILE MODEL: (PROBLEMS FACED IN AGILE MODEL)**

* Estimation of the cost at the starting of the project can be difficult( because you never know when and how many times the client can change the requirements)
* This model cost more compared to waterfall model
* More time required because the team needs to interact with the clients on a daily basis
* There is a huge dependency on the customers expectations and requirements. So if the customers are not giving a clear requirement and are going off track. It would be difficult got the development team and the team could go offtrack
* Due to lack of documentation, it makes it challenging for new team members to get onboard for future maintenance and support

**WHEN MIGHT YOU USE THE AGILE METHODOLOGY FOR THE SOFTWARE DEVELOPEMNT LIFE CYCLE PROCESS?**

* When the customers are not 100% clear on the requirements that might require changes in the requirement
* When clients want a daily interaction with the team
* When the client wants the software to be developed in a short period of time

**WHY DOES THE REQUIREMENT PHASE IN THE AGILE METHODOLOGY MATTER?**

The requirement gathering phase in the agile method is very crucial because, during the requirement phase, the stakeholders need to understand the needs and expectations of their client. This approach ensures that the final product meets the customer's actual requirements leading to customer satisfaction. The requirement phase usually involves creating user stories which help in prioritizing features based on the customers stories. The requirement phase also serves as the starting point for continuous changes throughout the development process.

**HOW DO YOU KNOW WHEN TO PROGRESS TO THE NEXT STAGE OF A PROJECT WHEN USING AGILE MODEL?**

In the agile model after every scrum, the team should assess whether the goals have been accomplished to another scrum. The team works on the user stories provided by the client. So in order to proceed to the next stage, they need to ensure that all planned user stories are completed, tested and are met with the acceptance criteria. And before proceeding to the next stage, the product increment that has been delivered must be reviewed and approved by the stakeholder.

**TEST DELIVERABLES**