|  |
| --- |
| * Unit Testing and its techniques * System Testing * Performance Testing * Regression Testing * User Acceptance Testing * Test Data Preparation * Introduction to Defects * Defect Logging and Tracking * Defect Tracking Parameters * Defect Life cycle Workflow |

**Unit Testing and its techniques**

* Unit Testing is defined as a type of software testing where individual components of a software are tested.
* Unit Testing of the software product is carried out during the development of an application.
* An individual component\module can either be an individual function or a procedure.
* In SDLC, Unit testing is the first level of testing done before integration testing. Unit testing is a type of testing technique that is usually performed by developers.

### **Objective of Unit Testing:**

The objective of Unit Testing is:

* To isolate a section of code.
* To verify the correctness of the code.
* To test every function and procedure.
* To fix bugs early in the development cycle and to save costs.
* To help the developers understand the code base and enable them to make changes quickly.
* To help with code reuse.

### **Workflow of Unit Testing:**

### A diagram of a diagram Description automatically generated

### **Unit Testing Techniques:**

There are 3 types of Unit Testing Techniques.

* **Black Box Testing:**
* Tester unaware of the internal structure
* Black Box Testing mainly focuses on input and output of software applications, based on software requirements and specifications.
* Testers can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.
* **White Box Testing:**
* Tester aware of the internal structure
* Testers can test the functional behaviour of the system by giving the input and checking the functionality output including the internal design structure and code of the modules.
* **Gray Box Testing:**
* Gray Box Testing is a software testing technique that is a combination of the Black Box Testing technique and the White Box Testing technique.
* This includes access to internal data structures and algorithms to design the test cases.
* This technique is used in executing the relevant test cases, test methods, and test functions, and analysing the code performance for the modules.

### **Unit Testing Tools:**

Here are some commonly used Unit Testing tools:

* Jtest
* Junit
* NUnit
* EMMA
* PHPUnit

### **Why Unit Testing is required:**

* Unit testing allows the programmer to refine code and make sure the module works properly.
* Unit testing enables testing parts of the project without waiting for others to be completed.
* **Early Detection of Issues:** Unit testing allows developers to detect and fix issues early in the development process before they become larger and more difficult to fix.
* **Improved Code Quality:** Unit testing helps to ensure that each unit of code works as intended and meets the requirements, improving the overall quality of the software.
* **Increased Confidence:** Unit testing provides developers with confidence in their code, as they can validate that each unit of the software is functioning as expected.
* **Faster Development:** Unit testing enables developers to work faster and more efficiently, as they can validate changes to the code without having to wait for the full system to be tested.
* **Better Documentation:** Unit testing provides clear and concise documentation of the code and its behaviour, making it easier for other developers to understand and maintain the software.
* **Reduced Time and Cost:** Unit testing can reduce the time and cost required for later testing, as it helps to identify and fix issues early in the development process.

**System Testing**

* **System Testing** is a level of testing that validates the complete and fully integrated software product.
* The purpose of a system test is to evaluate the end-to-end system specifications.
* System test involves the external workings of the software from the user’s perspective.
* Testing the fully integrated applications including external peripherals to check how components interact with one another and within the system. This is also called End to End testing scenario.

## Types of System Testing

* [**Usability Testing**](https://www.guru99.com/usability-testing-tutorial.html)**–** mainly focuses on the user’s ease to use the application, flexibility in handling controls and ability of the system to meet its objectives.
* [**Load Testing**](https://www.guru99.com/load-testing-tutorial.html)**–** is necessary to know that a software solution will perform under real-life loads.
* [**Recovery Testing**](https://www.guru99.com/recovery-testing.html)**–** is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes.
* [**Regression Testing**](https://www.guru99.com/regression-testing.html)**–** involves testing done to make sure none of the changes made over the course of the development process have caused new bugs. It also makes sure no old bugs appear from the addition of new software modules over time.
* **Migration Testing –** is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.
* **Functional Testing –** Also known as functional completeness testing,[Functional Testing](https://www.guru99.com/functional-testing.html) involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.

[**Examples of system testing in software testing**](https://www.bing.com/ck/a?!&&p=0f2eea51795abc4eJmltdHM9MTcxMjM2MTYwMCZpZ3VpZD0yM2U4ZDFlOS05ODJmLTYyZDAtM2Y3MC1jNWE5OTk4MjYzYTUmaW5zaWQ9NTkzOQ&ptn=3&ver=2&hsh=3&fclid=23e8d1e9-982f-62d0-3f70-c5a9998263a5&psq=system+testing+example+in+software+testing&u=a1aHR0cHM6Ly93d3cuc2NyaXB0d29ya3MuaW8vYmxvZy9ndWlkZS10by1zeXN0ZW0tdGVzdGluZy8&ntb=1)

* Testing a banking application for its functionality, security, and performance
* Testing an online airline’s booking system for browsing flight schedules and prices, selecting dates and times, etc.
* Testing an e-commerce company’s website for searching and filtering items, selecting an item, adding it to the cart, purchasing it, and more.

**Performance Testing**

* **Performance Testing** is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload.
* The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application.
* It is also known as “Perf Testing”.

The focus of Performance Testing is checking a software programs.

* Speed – Determines whether the application responds quickly.
* Scalability – Determines the maximum user load the software application can handle.
* Stability – Determines if the application is stable under varying loads.

**Regression Testing**

* **Regression Testing** is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
* A full or partial selection of already executed test cases that are re-executed to ensure existing functionalities work fine.
* Regression testing refers to the process of retesting the existing functionality of the code after making changes or enhancements to the code.
* It aims to ensure that the modifications or updates haven't introduced any new defects or issues, and that the existing functionality or the logics still works as expected.
* It ensures that the changes do not adversely impact the overall functionality of the code.

**Why Regression Testing?**

* Regression testing should be essential in the ETL testing process.
* As it can identify if code changes or enhancements are introducing new defects or disrupting existing functional tests.
* Without a regression testing process, even minor code changes may have a chance of leading to costly errors.

**When Regression Testing can be performed?**

Below are the few scenarios when you can apply the regression testing process.

**New functionality is added to the application:**

* This happens when new features or modules are created in an application or a website.
* The regression is performed to see if the existing features are working as usual with the introduction of the new feature.

**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 features that were there.

**After a defect is fixed:**

* The developers perform regression testing after fixing a bug in any functionality.
* This is done to determine if the changes made while fixing the bug have affected other related existing features.

**Once the performance issue is fixed:**

* After fixing any performance issues, the regression testing process is triggered to see if it has affected other existing functional tests.

**While integrating with a new external system:**

* End-to-end regression testing process is required whenever the product integrates with a new external system.

**How to do Regression Testing in Software Testing**

* As we discussed earlier, regression testing is performed based on any changes done to the existing code or application.
* It can be a bug fix, new feature integration, and so on.
* Whenever such code changes happen, the QA team performs the following activities given below.
* Discuss with the development team about the specific modules that were touched on during the code change.
* Discuss with the product owner about the change to the new feature and learn how it flows across or impacts other functionality.
* Identify the tests from the existing test suite that the testers need to execute to regress the existing features.

Various regression testing techniques can be carried out for effective software quality assurance:

A diagram of a process

Description automatically generated

### **Retest All**

* All the test cases in the existing test bucket or suite should be re-executed.
* This is a most expensive method as it requires a lot of time and resources to execute all the test cases.

### **Regression Test Selection**

* Regression Test Selection is a technique where some selected test cases from a test suite are executed.
* It helps test whether the modified code affects the software application or not.

### **Prioritization of Test Cases**

* Prioritization of the test cases depends on the business impact, criticality, and frequently used functional tests.
* It reduces the effort of executing regression tests.

## Selecting test cases for regression testing

An effective regression test suite can be built by selecting the following types of test cases –

* Test cases from functionalities/modules that have frequent defects.
* Functionalities that are more visible to the users
* Test cases which verify core features of the product
* Test cases of functionalities that have undergone more recent changes.
* All Integration test cases.
* All complex test cases.

## Types of Regression Testing

### **1) Unit Regression Testing (URT)**

* This is a very focused approach where only the modified section goes under the regression test instead of the impacted region.
* The other portions of the module remain unaffected.

**Example**

**As an** example, in Build 1, an issue was found and reported to the developer.

Let’s say it was a bug in the login functionality. So the developer fixes it, adds the bug fix in Build 2, and submits it. The testing team checks only if the login feature is working as expected instead of checking other features.

A screenshot of a login screen

Description automatically generated

### **2) Full Regression Testing (FRT):**

* This testing covers all the functionalities of an application. Full regression testing is usually performed in before final releases of a software product.
* In the second or third build, the customer or the business owner may ask for modifications. They may also demand new functionalities and or report defects.
* The testing team then conducts impact analysis, makes all the modifications, and performs a final complete product test.

For example, the 4th build is the final release before the launch. So, in this build, the testing team performs a complete test or retest of the product instead of just the impact area or a feature. This is done after the modifications and tests in builds 1, 2, and 3.

To perform complete regression testing,

* Changes are performed on the core components of the application. For example, if there is a modification core module of an application, then the whole application needs to be regressed.

**Test Cases for the Login Page:**

* Verify that the login page loads correctly and is accessible from the website's homepage.
* Check that the login credentials are case sensitive, and the appropriate message is displayed if the user enters incorrect information.
* Verify that the "Forgot Password" option works as intended, allowing users to reset their password in case they forget it.
* Ensure that the system limits the number of unsuccessful login attempts to prevent brute-force attacks.

**Test Cases for the Registration Page:**

* Verify that the registration page is accessible from the website's homepage and loads correctly.
* Check that the system validates the user's information, such as email address, phone number, and password complexity.
* Ensure that the system does not allow duplicate email addresses or phone numbers.
* Verify that the user receives an email or SMS confirmation after registering.

**Test Cases for the Ticket Booking Page:**

* Ensure that the ticket booking page displays accurate information about the event, such as the date, time, and venue.
* Verify that the system displays the total cost of the ticket purchase, including any taxes and fees.
* Ensure that the system accepts multiple payment options, such as credit/debit cards, PayPal, and mobile wallets.

**Test Cases for the Payment Gateway:**

* Verify that the payment gateway is secure and encrypts user information to prevent fraud.
* Check that the payment gateway accepts different currencies and displays the correct conversion rates.
* Ensure that the payment gateway sends a confirmation email or SMS to the user after the transaction is complete.

### **Partial Regression Testing:**

* Partial Regression Testing is used to verify that new code changes or enhancements do not impact existing functionality negatively.
* In partial regression testing, we focus only on specific parts of the software affected by the recent changes.
* Test cases for partial regression testing are carefully selected based on the impact analysis of the code changes.
* Identifying the correct test cases to include in the partial regression test suite is crucial.

**Automated Regression Testing**

* Automating regression tests is necessary when there are multiple releases.
* It is also required for multiple regression cycles and numerous repetitive activities.
* Performing multiple test cycles across releases is very time-consuming.

Below are the steps where automated regression testing is done:

**Step 1)** The manual testing team checks all requirements and identifies the impacted region. After this process, they forward the requirement test bundle to the automation team or automation engineer.

**Step 2)** The manual testing team begins testing the new modules while the automation test team writes the scripts and automates the test case.

**Step 3)** Before using this method of a regression test, the automation team identifies which test cases will support automation.

**Step 4)** They convert those regression tests into scripts depending on which test cases can be automated.

**Step 5)** During the scripting process, the automation team refers to the regression test cases. They don’t really require the product or the app knowledge.

**Step 6)** When the test scripts are completed, the automation team will execute them on the new application.

**Step 7)** After the execution, the result informs if the test was a Pass or Fail.

**Step 8)** If the test fails, it is re-checked using the manual testing method, and if the issue exists, it is reported to the respective developer.

Note: After the bug is fixed, the issue and the impact area are sent to the manual tester for retesting, and the automation team re-executes the script.

**Step 9)** This process continues until all the newly added regression features get a Pass status.

Here are the benefits of automated regression testing:

* **Reusable:** Its test scripts are reusable across multiple releases.
* **Accuracy:** The automation tools perform the task redundantly, reducing the chance of manual error.
* **Saves time:** It is faster than the manual functional testing process and is time efficient.
* **Batch execution:** It is possible to execute all the scripts simultaneously and parallelly in automated testing.
* **No resource increase required:** The regression test is bound to increase with every new release. However, you do not need to add new resources for automation.

## Difference between Retesting and Regression Testing

|  |  |
| --- | --- |
| **Retesting** | **Regression testing** |
| It is built specifically for defect fixes. | Regression testing is done mainly to verify if code changes have impacted other functionalities. |
| Retesting doesn’t check the other versions and only verifies if the affected functionalities are restored. | Focuses on previous versions, and it tests if the previous functions are still working as expected. |
| Each test is specific | Regression is a generic test. |
| This testing is for failed test cases. | It is for passed-test cases. |
| It checks specific defects, so it cannot be automated. | Can be automated as it invoves repeated execution on the test cases. |
| Retesting is not always a part of a testing cycle, as it is required only when bugs are found. | Regression is always a part of testing, as every time a code is changed, this test must be conducted to understand if the product functionality is stable. |
| It is high-priority testing as it focuses on known issues. | This is low-priority testing, as it is overall testing of possible defects. |
| This testing is not time-consuming as it works on a specific defect. | As it involves a large area of the software, hence, it is time-consuming. |
| It determines defects with the same data and environment with a different input and a new version. | This testing can acquire cases from user manuals, defect reports, and functional specifications. |

## User Acceptance Testing

* **User Acceptance Testing (UAT)** is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment.
* UAT is done in the final phase of testing after functional, integration and system testing done.

**SDLC Cycle-**

Development

Production deployment

UAT Phase

Testing

* The main Purpose of UAT is to validate end to end business flow.
* UAT users does not focus on cosmetic errors, spelling mistakes or system testing.
* User Acceptance Testing is carried out in a separate testing environment with production-like data setup.
* It is kind of black box testing where two or more end-users will be involved.

UAT is performed by –

* Client
* End users

## Need of User Acceptance Testing

* 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 the final product is accepted by client/end-user, user acceptance testing is needed.

### **Prerequisites of User Acceptance Testing**

Following are the entry criteria for User Acceptance Testing:

* Business Requirements must be available.
* Application Code should be fully developed.
* Unit Testing, Integration Testing & System Testing should be completed.
* No Showstoppers, High, Medium defects in System Integration Test Phase.
* Only Cosmetic error is acceptable before UAT.
* Regression Testing should be completed with no major defects.
* All the reported defects should be fixed and tested before UAT.
* Traceability matrix for all testing should be completed.
* UAT Environment must be ready.
* Sign off mail or communication from System Testing Team that the system is ready for UAT execution.

**Test Data Preparation**

* Test Data in Software Testing is the input given to a software program during test execution.
* Test data is the data used for the whole end to end testing process
* Test data is used for both positive testing to verify that functions produce expected results for given inputs and for negative testing to test software ability to handle unusual, exceptional or unexpected inputs.
* Poorly designed testing data may not test all possible test scenarios which will hamper the quality of the software.

## What is Test Data Generation? Why test data should be created before test execution?

* Everybody knows that testing is a process that produces and consumes large amounts of data.
* Depending on the testing environment test data can be created or at least identify a suitable test data for the test cases (is the test data is already created).
* Typically test data is created in-sync with the [test case](https://www.guru99.com/test-case.html) it is intended to be used for.

Test Data can be Generated –

* Manually
* Mass copy of data from production to testing environment
* Mass copy of test data from legacy client systems to the migrated system
* Automated Test Data Generation Tools

**Defect Analysis and Life Cycle**

**Introduction to Defects**

What Is a Defect?

A Defect, in simple terms, is a flaw or an error in an application that is restricting the normal flow of an application by mismatching the expected behaviour of an application with the actual one.

The defect occurs when any mistake is made by a developer during the designing or building of an application and when this flaw is found by a tester, it is termed as a defect.

It is the responsibility of a tester to do thorough testing of an application to find as many defects as possible to ensure that a quality product will reach the customer.

**Defect Management Tool – HPALM, Hp Quality Centre, IBM Rational Quality Manager**

**Defect Logging and Tracking**

**Defect logging-** It is a process of finding defects in the application under test or product by testing or recording feedback from customers and making new versions of the product that fix the defects or the client’s feedback.

**Defect tracking**- It is an important process in software engineering as Complex and business critical systems have hundreds of defects. One of the challenging factors is Managing, evaluating and prioritizing these defects. The number of defects gets multiplied over a period of time and to effectively manage them, defect tracking system is used to make the job easier.

**Defect Tracking Parameters**

Defect tracking Parameters,

* ID: A unique identifier for each defect.
* Title: A brief description of the defect.
* Description: A precise description of the defect, including the steps to reproduce the defect.
* Severity: The severity of the defect, such as critical, major, or minor.

A screenshot of a computer

Description automatically generated

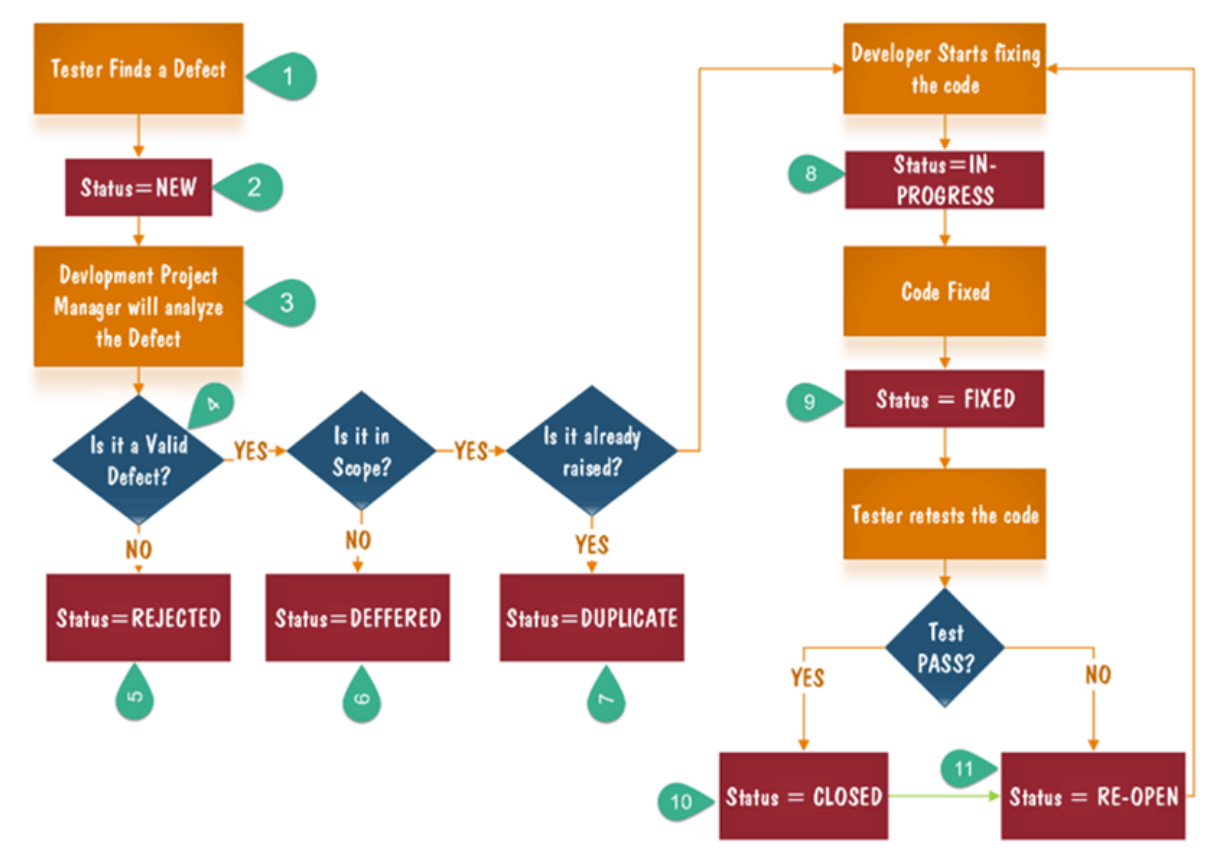
**Defect Life cycle workflow**

**Defect Life Cycle** or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate status of defect which changes to various assignees and make the defect fixing process systematic and efficient.

**Defect States Workflow**

The number of states that a defect goes through varies from project to project.

* **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
* **Open**: The developer starts analysing and works on the defect fix
* **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
* **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the [software testing](https://www.guru99.com/software-testing-introduction-importance.html) remains pending from the testers end, the status assigned is “pending retest.”
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”
* **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed, and the status assigned is “verified.”
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again, the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status “Closed.”
* **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs
* **Not a bug:** If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

****

1. Tester finds the defect
2. Status assigned to defect- New
3. A defect is forwarded to Project Manager for analyse
4. Project Manager decides whether a defect is valid
5. Here the defect is not valid- a status is given “Rejected.”
6. So, project manager assigns a status rejected. If the defect is not rejected, then the next step is to check whether it is in scope. Suppose we have another function- email functionality for the same application, and you find a problem with that. But it is not a part of the current release when such defects are assigned as a postponed or deferred status.
7. Next, the manager verifies whether a similar defect was raised earlier. If yes defect is assigned a status duplicate.
8. If no the defect is assigned to the developer who starts fixing the code. During this stage, the defect is assigned a status in- progress.
9. Once the code is fixed. A defect is assigned a status fixed
10. Next, the tester will re-test the code. In case, the[Test Case](https://www.guru99.com/test-case.html)passes the defect is closed. If the test cases fail again, the defect is re-opened and assigned to the developer.
11. Consider a situation where during the 1st release of Flight Reservation a defect was found in Fax order that was fixed and assigned a status closed. During the second upgrade release the same defect again re-surfaced. In such cases, a closed defect will be re-opened.