**UNIT-2**

**Software Testing Life Cycle**

**Syllabus:**

Software Testing Life Cycle: Basic Concepts of Software Testing Life Cycle, Testing Methodologies, Test Plans, Test Cases, Test Executions and Defect Reports. Defects: Types of Defects, Defect Life Cycle, Levels vs Builds, Priority and Severity. Types of Testing: Functionality Testing, Security Testing, Smoke Testing, Sanity Testing, Adhoc Testing, Exploratory Testing, Load Testing, Stress Testing, Regression Testing, Retesting.

**Software Testing Life Cycle:**

The Software Testing Life Cycle (STLC) is a systematic approach to testing a software application to ensure that it meets the requirements.

“STLC consists of series of activities carried out methodologically to help certify your software product. These activities are part of the Software Testing Life Cycle.”

The different stages in Software Test Life Cycle:

1. Requirement Analysis
2. Test Planning
3. Test case Development
4. Test Environment Setup
5. Test Execution
6. Test Closure
   1. **Requirement Analysis: “**During this phase, [test team](http://www.guru99.com/how-to-organize-a-test-team.html) studies the requirements from a testing point of view to identify the testable requirements.”

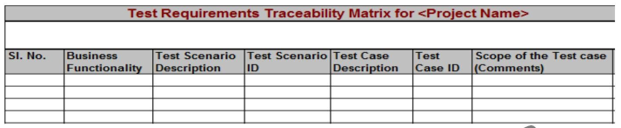
The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, and System Architects etc.) to understand the requirements in detail. Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance[/security](http://www.guru99.com/ethical-hacking-tutorials.html) availability).

**Deliverables:** Requirement Traceability Matrix (RTM)

**RTM: “**Traceability matrix links a business requirement to its corresponding functional requirement.”

If a Test Case fails, traceability helps determine the corresponding functionality easily; it also helps ensure that all requirements are tested.

#### Sample RTM:



* 1. **Test Planning:**

**What is Test plan:** “Test planning is the first step of the testing process. In this phase we identify the activities and resources which would help to meet the testing objectives.”

**What Test Plan contains: (IEEE 829 STANDARD TEST PLAN TEMPLATE)**

Test plan identifier

Test deliverables

Introduction

Test tasks

Test items

Environmental needs

Features to be tested

Responsibilities

Features not to be tested

Staffing and training needs

Approach Schedule

Item pass/fail criteria

Risks and contingencies

**Who will prepare the test plan:** Test Lead or Test Manager.

**What is Test Strategy:**“A Test Strategy document is a high level document and normally developed by project manager. This document defines “Software Testing Approach” to achieve testing objectives. The Test Strategy is normally derived from the Business Requirement Specification document.”

**Test plan Vs Test Strategy:** Generally it doesn’t matter which comes first. Test planning document is a combination of strategy plugged with overall project plan. According to IEEE Standard 829-2008, strategy plan is a sub item of test plan.

Every organization has their own standards and processes to maintain these documents. Some organizations include strategy details in test plan itself. Some organizations list strategy as a subsection in testing plan but details is separated out in different test strategy document.

**Ex:**Test plan gives the information of who is going to test at what time. For example: Module 1 is going to be tested by “X tester”. If tester Y replaces X for some reason, the test plan has to be updated.

On the contrary, test strategy is going to have details like – “Individual modules are to be tested by test team members. “ In this case, it does not matter who is testing it- so it’s generic and the change in the team member does not have to be updated, keeping it static.

**Deliverables:** Test Plan with estimation

* 1. **Test Case Development:**

**“**During this phase the test cases will be prepared”.

**2.3.1 What is test case**: “A test case is a set of conditions under which a tester will determine whether a system under test satisfies requirements or works correctly.”

#### Writing Good Test Cases:

* + - * As far as possible, write test cases in such a way that you test only one thing at a time. Do not overlap or complicate test cases. Attempt to make your test cases ‘atomic’.
      * Ensure that all positive scenarios and negative scenarios are covered.
      * Language:

Write in simple and easy to understand language.

Use active voice: Do this, do that.

Use exact and consistent names (of forms, fields, etc).

* + - * Characteristics of a good test case:

Accurate: Exacts the purpose.

Economical: No unnecessary steps or words.

Traceable: Capable of being traced to requirements.

Repeatable: Can be used to perform the test over and over.

Reusable: Can be reused if necessary.

**2.3.2 Test Case Techniques:**

* + - * Equivalence Portioning (EP)
      * Boundary Value Analysis (BVP)

### Error Guessing

**Equivalence Portioning:**“EP divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.”

Test cases are designed to cover each partition at least once. This technique tries to define test cases that uncover classes of errors, thereby reducing the total number of test cases that must be developed. An advantage of this approach is reduction in the time required for testing a software due to lesser number of test cases.

**Boundary Value Analysis:**“Tests are designed to include representatives of boundary values “

**Error Guessing:**“The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.”

The success of error guessing is very much dependent on the skill of the tester, as good testers know where the defects are most likely to be. This is why an error guessing approach, used after more formal techniques have been applied to some extent, can be very effective.

#### 2.3.3 Test Data Preparation

**What is Test Data: “**In order to test a software application you need to enter some data for testing most of the features. Any such specifically identified data which is used in tests is known as test data.”

**Test data preparation:** In the above example we can generate the inputs for Valid and Invalid partitions.

#### Ex:

Valid: 1, 2, 5, 10, 11, 12

Invalid: -2, -1, 0, 13, 15, 99, 150, 999

#### 2.3.4 Types of test cases

**Functional Test Cases:** “The test cases based on functional requirement specifications”

**Positive Test Cases:** “Test Cases with valid input and also verifying that the outputs are correct.”

**Negative Test Cases:**“This testing involves exercising application functionality using a combination of invalid inputs, some unexpected operating conditions and by some other “out-of-bounds” scenarios.”

**Non Functional Test Cases:**“The test cases based on functional requirement specifications like performance, Load, Stress, Security, etc.”

**2.3.5 Test Case Review:** Reviewing is a form of testing too – the verification part of the V&V, also called static testing.

**Why Review:** For exactly the same reason we test the software

* + - * To uncover errors
      * To check for completeness
      * To make sure the standards and guidelines are followed

#### Review Checklist:

* + - * Do test cases cover all requirements?
      * Has each test case been assigned a test case identifier?
      * Does each test case specify?
        + Actions
        + Test condition
        + Expected result
      * Have the expected results been recorded in detail?
      * Is any method for validating expected results specified?
      * Do test cases for field validations, record validations and database updates include the following?
        + Valid conditions
        + Invalid conditions
        + Boundary or unusual conditions
      * Do the test cases for reports include the test data along with the expected output?
      * Have the inter test case dependencies been described?
      * Have Pass/Fail criteria been specified?
      * Have all requested environments been specified?
      * Has the method for logging on to the test environment been specified?
      * Are pre-conditions for the test specified?
      * Is the number of Test cases met customer standards?

**Self-Review: “**Review our own work by us”

**Peer Review:** “Review our own work by colleague”

**Lead Review:** “Review our work by our Test lead or Manager”

**Client Review:** “Review our work by client or business team after lead review.”

**Deliverables:** Test Case Document, Test Data

## **Test Environment Setup**

“Test environment decides the software and hardware conditions under which a work product is tested. “

Test environment set-up is one of the critical aspects of testing process and can be done in parallel with Test Case Development Stage. [Test team](http://www.guru99.com/how-to-organize-a-test-team.html) may not be involved in this activity if the customer or dev team provides the test environment in which case the [test team](http://www.guru99.com/how-to-organize-a-test-team.html) is required to do a readiness check (smoke testing) of the given environment.

**Deliverables:** Test Environment

**How many environments do we have:** “A Typical project can have following environments”

* Dev
* QA
* Pre-Production
* Production

## **Test Execution**

“In this phase testing team start executing test cases based on prepared test planning & prepared test cases in the prior step in testing environment”.

**Test Execution Activities**

* Execute tests as per plan
* Document test results, and log defects for failed cases
* Map defects to test cases in RTM
* Retest the Defect fixes
* Track the defects to closure

**Deliverables:** Completed RTM with the execution status, Test cases updated with results, Defect reports

## **Test Closure:**

**Test Cycle Closure Activities**

* Evaluate cycle completion criteria based on Time, Test coverage, Cost,Software, Critical Business Objectives, Quality
* Prepare test metrics based on the above parameters.
* Prepare Test closure report
* Qualitative and quantitative reporting of quality of the work product to the customer.
* Test result analysis to find out the defect distribution by type and severity.

**Deliverables:** Test Closure report, Test metrics

**Severity:**It is the extent to which the [defect](http://istqbexamcertification.com/what-is-defect-or-bugs-or-faults-in-software-testing/) can affect the software. In other words it defines the impact that a given defect has on the system.

If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

* **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
* **Major:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
* **Medium:** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.
* **Minor:** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
* **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and feel of the application then the severity is stated as cosmetic.

**Priority:** “Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait?”

This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements.

**For example:** If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

* Low: The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.
* Medium: The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
* High: The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.

#### Few very important scenarios related to the severity and priority :

* **High Priority & High Severity:** An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn’t allow to save the record then this is high priority and high severity bug.)
* **High Priority & Low Severity:** The spelling mistakes that happens on the cover page or heading or title of an application.
* **High Severity & Low Priority:** An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.
* **Low Priority and Low Severity:** Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

#### Famous Defect Tracking Tools:

* Bugzilla
* JIRA
* ALM (QC)

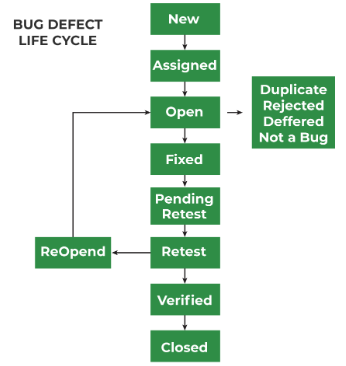
#### A typical Defect Tracker will have:

* Defect id
* Date
* Created By
* Assigned TO
* Bug description
* Steps to Reproduce
* Expected Result
* Comments
* Screen shots

**Defect Life Cycle:**“Defect life cycle is a cycle which a defect goes through during its lifetime.”

It starts when defect is found and ends when a defect is closed, after ensuring it’s not reproduced. [Defect life cycle](http://istqbexamcertification.com/what-is-a-defect-life-cycle/) is related to the bug found during testing.

The Life cycle of the bug can be shown diagrammatically as follows:



* **New:** When a defect is logged and posted for the first time. It’s state is given as new.
* **Assigned:** After the tester has posted the bug, the lead of the tester approves that the bug is genuine and he assigns the bug to corresponding developer and the developer team. It’s state given as assigned.
* **Open:** At this state the developer has started analyzing and working on the defect fix.
* **Fixed:** When developer makes necessary code changes and verifies the changes then he/she can make bug status as ‘Fixed’ and the bug is passed to testing team.
* **Pending retest:** After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.
* **Retest:** At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.
* **Verified:** The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to “verified”.
* **Reopen:** If the bug still exists even after the bug is fixed by the developer, the tester changes the status to “reopened”. The bug goes through the life cycle once again.
* **Closed:** Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to “closed”. This state means that the bug is fixed, tested and approved.
* **Duplicate:** If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to “duplicate“.
* **Rejected:** If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to “rejected”.
* **Deferred:** The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are priority of the bug may be low, lack of time for the release or the bug may not have major effect on the software.
* **Not a bug:** The state given as “Not a bug” if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and feel of the application like change of color of some text then it is not a bug but just some change in the looks of the application.

## **Metrics:**

A Metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute.

Metrics can be defined as “STANDARDS OF MEASUREMENT”.

Software Metrics are used to measure the quality of the project. Simply, Metric is a unit used for describing an attribute. Metric is a scale for measurement.

Suppose, in general, “Kilogram” is a metric for measuring the attribute “Weight”. Similarly, in software, “How many issues are found in thousand lines of code?”, here No. of issues is one measurement & No. of lines of code is another measurement. Metric is defined from these two measurements.

**Test metrics example:**

* How many defects are existed within the module?
* How many test cases are executed per person?
* What is the Test coverage %?

## Why Test Metrics?

Generation of Software Test Metrics is the most important responsibility of the Software Test Lead/Manager.

**“We cannot improve what we cannot measure.”**

**“We cannot control what we cannot measure”**

* Take decision for next phase of activities
* Evidence of the claim or prediction
* Understand the type of improvement required
* Take decision on process or technology change

**Effectiveness:** Doing the right thing. It deals with meeting the desirable attributes that are expected by the customer.

**Efficiency:** Doing the thing right. It concerns the resources used for the service to be rendered

1. **Test Plan coverage on Functionality**:

Formula: (No of requirements covered / total number of requirements) \* 100

1. **Test Case defect density**:

Formula: (Defective Test Scripts /Total Test Scripts) \* 100

**Example:** Total test script developed 1360, total test script executed 1280, total test script passed 1065, total test script failed 215.

So, test case defect density is :215 X 100 / 1280 = 16.8%

This 16.8% value can also be called as test case efficiency %, which is depends upon total number of test cases which uncovered defects.

1. **Defect Slippage Ratio:** Number of defects slipped (reported from production) v/s number of defects reported during execution.

Formula: Number of Defects Slipped / (Number of Defects Raised - Number of Defects Withdrawn)

Example: Customer filed defects are 21, total defect found while testing are 267, total number of invalid defects are 17.

So, Slippage Ratio is [21/ (267-17)] X 100 = 8.4%

**Build Vs Release:**

**Build**:

After developing the software module, developers convert the source codes into a standalone form or an executable code. Then the development team hands over the build to the testing team to perform testing. Build is in the testing phase; it may have already undergone testing or not. Software testing team checks this build. If it consists of multiple bugs and if it does not meet the requirements, then the software testing team rejects that build.  Builds occur prior to the release, and they are generated more frequently.

**Release**:

The release is the final application after completing development and testing. After testing the build, the testing team certifies that software and delivers it to the customer. It is possible for one release to have several builds. Therefore, it is the software delivered to the customer after completing the development and testing phases. Moreover, the release is based on builds, and it can have several builds.

**Types of Testing:**

**Functional Testing:**

**“**Functional Testing is a testing technique that is used to test the features/functionality of the Software”.

Theseare the following types of functional testing.

1. **Unit Testing:** Unit testing is a type of software testing, where the individual unit or component of the software tested. Unit testing, examine the different part of the application, by unit testing functional testing also done, because unit testing ensures each module is working correctly.

The developer does unit testing. Unit testing is done in the development phase of the application.

**2)** **Integration Testing:** Integration testing combined individual units and tested as a group. The purpose of this testing is to expose the faults in the interaction between the integrated units.

Developers and testers perform integration testing.

1. **Smoke Testing:**“Smoke testing refers to testing the basic functionality of the build declared as unstable and it is NOT tested anymore until the smoke test of the build passes.” Smoke testing is known as "**Build Verification Testing**."
2. **Sanity Testing:“**Software testing technique performed by the test team for some basic tests. The aim of basic test is to be conducted whenever a new build is received for testing. “

The terminologies such as Smoke Test or Build Verification Test or Basic Acceptance Test or Sanity Test are interchangeably used, however, each one of them is used under a slightly different scenario.

Sanity test is usually unscripted, helps to identify the dependent missing functionalities. It is used to determine if the section of the application is still working after a minor change. Sanity testing can be narrow and deep. Sanity test is a narrow regression test that focuses on one or a few areas of functionality

1. **Re-testing: “**Retesting is executing a previously failed test against new software to check if the problem is resolved. “
2. **Regression Testing: “**Regression testing is performed to verify if the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement. **“**

The purpose of a regression testing is to verify that modifications in the software or the environment have not caused any unintended adverse side effects and that the system still meets its requirements.

1. **Exploratory Testing:** “Testing of software without any documents (test cases or test planning) and Identify the functionality of application by exploring the application.
2. **Monkey Testing:** “Monkey testing is a software testing technique in which the testing System under test randomly. The Input data that is used to test also generated randomly.
3. **End to End Testing:**“End-to-end testing is a methodology used to test whether the flow of an application is performing as designed from start to finish.”

The purpose of carrying out end-to-end tests is to identify system dependencies and to ensure that the right information is passed between various system components and systems.

1. **Ad-hoc testing:**Ad-hoc testing is an informal testing type whose aim is to break the system. This type of software testing is unplanned activity. It does not follow any test design to create the test cases. Ad-hoc testing is done randomly on any part of the application; it does not support any structured way of testing.

Various ways to make Adhoc Testing More Effective

1. Preparation: By getting the defect details of a similar application, the probability of finding defects

in the application is more.

2. Creating a Rough Idea: By creating a rough idea in place the tester will have a focussed approach.

It is NOT required to document a detailed plan as what to test and how to test.

3. Divide and Rule: By testing the application part by part, we will have a better focus and better

understanding of the problems if any.

4. Targeting Critical Functionalities: A tester should target those areas that are NOT covered while

designing test cases.

5. Using Tools: Defects can also be brought to the lime light by using profilers, debuggers and even

task monitors. Hence being proficient in using these tools one can uncover several defects.

6. Documenting the findings: Though testing is performed randomly, it is better to document the tests

if time permits and note down the deviations if any. If defects are found, corresponding test cases

are created so that it helps the testers to retest the scenario.

**Non-Functional Testing:**

Non-Functional testing is a software testing technique that checks the non-functional attributes of the system.



1. **Security Testing:** “Security testing is basically to check that whether the application or the product is secured or not. “

Anyone can come tomorrow and hack the system or login the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended.

Security testing is related to the security of data and the functionality of the application. You should be aware of the following concepts while performing security testing:

1. Confidentiality - The application should only provide the data to the relevant party e.g. one customer's transactional data should not be visible to another customer; the irrelevant personal details of the customer should not be visible to the administrator and so on.
2. Integrity - The data stored and displayed by the application should be correct e.g. after a withdrawal, the customer's account should be debited by the correct amount.
3. Authentication - It should be possible to attribute the data transmitted in the application to either the application or the customer. In other words, no one other than the customer or the bank should be able to create or modify any data.
4. Authorization - The application or a user should only be able to perform the tasks which they are respectively authorized to perform e.g. a customer should not be able to withdraw more than the balance in their account without having an overdraft facility, the application should not be able to levy charges on a customer account without prior customer approval.
5. Availability - The data and functionality should be available to the users throughout the working period e.g. if the bank's operating times are from 8 a.m. to 8 p.m. on all working days, it should be possible for a customer to access their account and make the necessary transactions on their account.
6. Non-repudiation - At a later date, it should not be possible for a party to deny that a particular transaction or data change took place e.g. if a customer withdraws an amount from their account, this should trigger the relevant actions (posting to their transaction records, debiting their account and sending them a notification etc.).
7. **Graphical User Interface Testing:**“Graphical User Interface (GUI) testing is checking the application design of an application”.

Ex: Required/Optional, Fields Align, Lengths, Progress Bars, Scroll Bars, Alignments, etc

1. **Usability Testing:** “In usability testing basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application is user-friendly or not. “

Usability Testing tests the following features of the software.

* How easy it is to use the software.
* How easy it is to learn the software.
* How convenient is the software to end user.

1. **Stress Testing:** “It is a form of testing that is used to determine the stability of a given system, Stress testing involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. “

Stress testing is a generic term used to describe the process of putting a system through stress.

1. **Load Testing:** “Load testing is performed to determine a system’s behavior under both normal and at peak conditions. “

A load test is usually conducted to understand the behavior of the application under a specific expected load.

E.g. If the number of users are increased then how much CPU, memory will be consumed, what is the network and bandwidth response time.

1. **Performance Testing:** “Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. “

It can serve different purposes like it can demonstrate that the system meets performance criteria.

1. **Localization Testing:** “Localization translates the product UI and occasionally changes some initial settings to make it suitable for another region.” Localization testing checks the quality of a product's localization for a particular target culture/locale.

The test effort during localization testing focuses on:

* Areas affected by localization, such as UI and content
* Culture/locale-specific, language-specific, and region-specific areas

1. **Globalization Testing:** “Globalization Testing is testing process to check whether software can perform properly in any locale or culture & functioning properly with all types of international inputs and steps to effectively make your product truly global.”

This type of testing validates whether the application is capable for using all over the world and to check whether the input accepts all the language texts.

Ex: Let’s see another example of a Zip code field in Sign up form:

* For globalized, it should allow to enter alphanumeric inputs
* For localized (country like INDIA), it should allow only numbers in input field.

1. **Compatibility Testing:** ”Compatibility Testing ensure compatibility of the application built with various other objects such as other web browsers, hardware platforms, operating systems etc.”

This type of testing helps find out how well a system performs in a particular environment that includes hardware, network, operating system and other software etc.

Ex: Browser Compatibility Testing, OS Compatibility Testing

1. **Installation Testing:** “Installation testing is performed to ensure that all necessary components are installed properly and working as per the requirements of the software, post installation. Installation process may include partial, full or upgrade install. “
2. **Recovery Testing:** “Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or failure“

Ex: For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.

**Types of Defects:**

Depending on the nature of defects, they are classified into:

**Functional Defects:** Functional bugs are those which cause the software to malfunction. A good example of this would be a button which, when clicked, is supposed to open a new window – but instead, nothing happens. Functional bugs can be fixed by [performing functional testing](https://www.softwaretestingmaterial.com/functional-testing/).

**Unit Level Bugs:** Unit level bugs are defects that are related to the functionality of a single software unit. A software unit is the smallest testable part of an application. Examples of software units include classes, methods, and procedures. Unit level bugs can have a significant impact on the overall quality of the software.

Unit level bugs can be fixed by performing Unit testing.

**Integration Level Bugs:** Integration level bugs are defects that occur when two or more software units are combined. These defects can be difficult to find and fix because they often require coordination between multiple teams. However, they can have a significant impact on the overall quality of the software.

Integration bugs can be fixed by performing integration testing.

**Usability defects:** Usability bugs are defects that impact the user experience of the software that makes it difficult to use. A usability defect is a defect in the user experience of software that makes it difficult to use. Usability bugs are the bugs such as If a website is complicated to access or get around or the signup process is complicated to go through.

During usability testing, software testers check apps against user requirements and Web Content Accessibility Guidelines (WCAG) to look for such problems. However, they can have a significant impact on the overall quality of the software.

Usability bugs can be fixed by performing usability testing.

**Performance defects:** Performance bugs are defects that impact the performance of the software. This can include things like the speed of the software, how much memory it uses, or how many resources it consumes. Performance level bugs can be difficult to track down and fix, because they can be caused by a number of different factors.

Usability bugs can be fixed by performing Performance testing.

**Security defects:** Security bugs are a type of software defect that can have major consequences if left unaddressed. These defects can allow malicious users to gain access to sensitive data or systems, or even allow them to take control of the affected software. As such, it is critical that security level bugs are given high priority and addressed as soon as possible.

Security bugs can be fixed by performing Security testing.

**Compatibility defects:** Compatibility defects are those bugs which occur when an application is not compatible with the hardware it is running on, or with other software it needs to interact with. Incompatibility between software and hardware can result in crashes, data loss, and other unpredictable behavior. Testers need to be aware of compatibility issues and test accordingly. A software application that has compatibility issues does not run consistently on different sorts of hardware, operating systems, web browsers, and devices when connected with certain programs or running under specific network conditions.

Compatibility bugs can be fixed by performing Compatibility testing.

**Regression Defects:** A regression defect occurs when a code change causes an unintended effect on an independent part of the software.

Regression bugs can be fixed by performing Regression testing.

**principles of software testing: -**

Software testing is governed by **seven principles**:

* **Absence of errors fallacy:**Even if the software is 99% bug-free, it is unusable if it does not conform to the user's requirements. Software needs to be bug-free 99% of the time, and it must also meet all customer requirements.
* **Testing shows the presence of errors:** Testing can verify the presence of defects in software, but it cannot guarantee that the software is defect-free. Testing can minimize the number of defects, but it can't remove them all.
* **Exhaustive testing is not possible:**The software cannot be tested exhaustively, which means all possible test cases cannot be covered. Testing can only be done with a select few test cases, and it's assumed that the software will produce the right output in all cases. Taking the software through every test case will cost more, take more effort, etc., which makes it impractical.
* **Defect clustering:** The majority of defects are typically found in a small number of modules in a project. According to the Pareto Principle, 80% of software defects arise from 20% of modules.
* **Pesticide Paradox:**It is impossible to find new bugs by re-running the same test cases over and over again. Thus, updating or adding new test cases is necessary in order to find new bugs.
* **Early testing:** Early testing is crucial to finding the defect in the software. In the early stages of SDLC, defects will be detected more easily and at a lower cost. Software testing should start at the initial phase of software development, which is the requirement analysis phase.
* **Testing is context-dependent:**The testing approach varies depending on the software development context. Software needs to be tested differently depending on its type. For instance, an ed-tech site is tested differently than an Android app.