**Unit - 3: Levels of Testing**

**3.1 Levels of Testing:**

* Testing is a life-cycle activity which starts at the time of proposal and ends when acceptance testing is completed and product is finally delivered to customer.
* Testing begins with a proposal for software/system application development/maintenance, and ends when the system is formally accepted by the user/customer.
* Different agencies/stakeholders are involved in conducting
* specialised testing required for the specific application.
* Definition of these stakeholders/agencies depends on the type of testing involved and level of testing to be done in various stages of the software development life
* cycle.
* Today we will learn the level of testing performed by different agencies.
* which may differ from customer to customer, product to product and organisation to organisation.

**3.2 Proposal Testing:**

* A proposal is made to the customer on the basis of **Request for Proposal (RFP) or Request for Information (RFI) or Request for Quotation (RFQ)** by the customer
* Before making any proposal, the supplier **must understand the purpose of such request**, and devise the proposed solution accordingly
* One must **understand customer problem and the possible solution**.
* Any proposal prepared in response to such request is reviewed by an organisation before sending it to the customer.
* It is **reviewed by different panels or groups** in the of organisation such as technical group and commercial group.

***Technical review****:-* Technical review mainly involves technical feasibility of the kind of system or application proposed, the availability and requirement of skill sets, the hardware/software and other requirements of system.

***Commercial Review***:- A proposal undergoes financial feasibility and other types of feasibilities involved with respect to the business

**3.3 Requirement Testing:**

* Requirement creation involves gathering customer requirements and arranging them in a form to verify and validate them.
* The requirements may be categorized into different types such as technical, economical, legal, operational and system requirements.

***Validation of requirements:*** Requirement testing involves writing use case using requirement statements.

No assumption is to be made while writing use cases from requirement statements.

Requirements can be considered as clear, complete, measurable, testable and not conflicting with each other.

Requirements testing is done to clarify whether project requirements are feasible or not in terms of time, resources and budget.

Many bugs emerge in software because of incompleteness, inaccuracy and ambiguities in functional requirements.

That’s why it is highly important to test requirements and eliminate ambiguities before you start to develop a project.

This type of testing covers testing of requirements specification that describes:

* project functionality
* user interface
* software and hardware interfaces
* performance criteria
* implementation issues and risks
* security and system correctness criteria

**3. 4 Design Testing:**

* Once the requirement statement satisfies all characteristics defined above, the system architect starts with system high-level architectural designing.
* The designs made by system architects must be traceable to requirements.

The design must also possess the characteristics given below.

***Clarity*:** A design must define all functions, components, tables, stored procedures, and reusable components very clearly.

***Complete*:** A design must be complete in all respect. It must define the parameters to be passed/received, formats of data handled, etc.

***Traceable*:** A design must be traceable to requirements. The project manager must check if there is any requirement which does not have corresponding design or vice versa.

***Implementable:*** A design must be made in such a way that it can be implemented easily with selected technology and system.

It must guide the developers in coding and compiling the code.

***Testable:*** Testers make structural test cases on the basis of design Thus, a good design must help testers in creating structural test cases

**3.5 Code Review:**

* Code reviews include reviewing code files, database schema, classes, object definitions, procedures, and methods
* Code review is applied to ensure that the design is implemented correctly by the developers, and guidelines and standards available for the purpose of coding are followed correctly.

**Characteristics:**

***Clarity* :** Code must be written correctly as per coding standards and syntax, requirements for the given platform. It must follow the standards and guidelines defined by the organisation/project/customer, as the case may be.

***Complete:*** Code, class, procedure, and method must be complete in all respect. It must suffice the purpose for which it is created.

***Traceable:*** Code must be traceable with design components. It must declare clearly about the requirements, design, author, date, etc.

***Maintainable:*** Code must be maintainable Any developer with basic knowledge and training about coding must be able to handle the code in future while maintaining or bug fixing it.

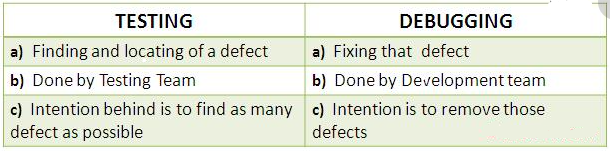
**3.6 Unit Testing:**

Unit is the smallest part of a software system which is testable.

It may include code files, classes and methods which can be tested individually for correctness.

Unit testing is a validation technique using black box methodology.

* **Individual components and units are tested** to ensure that they work correctly as an individual as defined in design.
* Unit testing may be **performed in debugger mode** to find how the variable values are changed during the execution.



**3.7 Module Testing:**

Many units come together and form a module.

If the module can work independently, it is tested by tester.

Module testing mainly concentrates on the structure of the system.

Module testing is done on related unit-tested components to find whether individually tested units can work together as a module or not. Module tests, often referred to as unit tests or component tests, are used to check the individual components of computer programs. With these tests, you can examine whether individual parts are functioning correctly before they are properly integrated into the overall software concept. These tests are useful, as they enable you to quickly and easily check whether the components are working as the developer intended them to. Module tests are one of the most effective ways to detect as many **errors as possible in the earlier stages of the program code**.

In general, module tests play an important role in the **quality assurance of software**. This method is particularly used in agile software development, where the goal is to carry out a lot of component tests. This is also the reason they are performed automatically. Corresponding programs can execute a number of module tests at the push of a button. Usually, the test program that’s used is written in the same language as the test object itself.

**3.8 Integration Testing:**

A type of software testing in which the **different units, modules or components of a software application are tested** as a combined entity

Integration testing involves integration of units to make a **module/integration of modules** to make a system/integration of a system with environmental variables if required to create a real-life application.

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

**3.9 Big-Bang Testing:**

Big-bang approach is the most commonly seen approach at many places, where the **system is tested completely after development is over**.

There is **no testing of individual units/modules and integration** sequence.

Sometimes, it includes huge amount of random testing which may not be repeatable.

**Advantages of Big Bang:**

* The advantage of Big Bang is that it’s very simple and easy to implement.
* This model requires very little or no planning.
* There is no formal procedure are required before starting of any project so this model is easy to manage.
* It is ideal for repetitive or small projects with minimum risks.

**Disadvantages of Big Bang Model:**

* Due to there is no pre planning required before starting the project hence the Big Bang model is a very high risky model.
* In addition if changes in the requirements or misunderstood requirements may even lead to complete reversal or scraping of the project.

**3.10 Sandwich Testing:**

* **Sandwich Testing** is the combination of bottom-up approach and top-down approach, so it uses the advantage of both bottom up approach and top down approach.

**Advantage of Sandwich Testing:**

* Sandwich Testing approach is used in very large projects having sub projects.
* It allows parallel testing.
* Sandwich testing is time saving approach.
* sandwich testing performs more coverage with same stubs.

**Disadvantage of Sandwich Testing:**

* Sandwich Testing is very costly.
* Sandwich Testing can not be used for such systems which have a lot of interdependence between different modules.
* In sandwich testing the need of stubs and drivers is very high.

**Stubs:**

**A stub** is a replica of a module that collects the data and develops many possible data. However, it executes like an actual module and is mainly used to test modules.

**Drivers:**

The **Drivers** establish the test environments and takes care of the communication, estimates results, and also sends the reports.

**There may be a question coming up in your mid saying   
difference between Stubs and Drivers.**

