**Testing**

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified?. Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Validation : Are we doing the right job?

Verification : Are we doing the job right?

Software testing should not be confused with debugging. Debugging is the process of analyzing and localizing bugs when software does not behave as expected. Although the identification of some bugs will be obvious from playing with the software, a methodical approach to software testing is a much more thorough means for identifying bugs. Debugging is therefore an activity which supports testing, but cannot replace testing.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis

looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted System atically. Testing begins at the module level and work towards the integration of entire computers based System . Nothing is complete without testing, as it vital success of the System testing objectives, there are several rules that can serve as testing objectives. They are

• Testing is a process of executing a program with the intend of finding an error.

• A good test case is one that has high possibility of finding an undiscovered error.

• A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncovered errors in the software also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

• For correctness

• For implementation efficiency

• For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

**1. TEST PLAN:**

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

* Unit Testing
* Integration Testing
* System Testing
* Data Validation Testing
* Output Testing

**1.1. UNIT TESTING:**

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. Unit testing, also known as component testing , refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

**1.2. INTEGRATION TESTING:**

Integration testing is System atic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

After unit testing in Sell-Soft System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

**1.3. SYSTEM TESTING:**

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

**1.4. DATA VALIDATION TESTING:**

This is the final step in testing. In this the entire System was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

**1.5. OUTPUT TESTING:**

The System considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with perspective System ; user at the time of developing and making changes whenever required.

This done with respect to the following points:

* Input Screen Designs,
* Output Screen Designs, Online message to guide the user and the like.The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the System testing. After preparing the test data, the System under study is tested using that test data. While testing the System by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**2. IMPLEMENTATION PROCEDURES:**

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the System . In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that

• The active user must be aware of the benefits of using the new System.

• Their confidence in the software is built up.

• Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the System, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won’t take place.

**2.1. USER TRAINING:**

User training is designed to prepare the user for testing and converting the System . To achieve the objective and benefits expected from computer based System , it is essential for the people who will be involved to be confident of their role in the new System . As System becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

**2.2. TRAINING ON THE APPLICATION SOFTWARE:**

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new System such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the System or part of the System while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

**2.3. OPERATIONAL DOCUMENT:**

Once the implementation plan is decided, it is essential that the user of the System is made familiar and comfortable with the environment. Education involves right atmosphere and motivating the user. A documentation providing the whole operations of the System is being developed in such a way that the user can work with it in well consistent way. The System is developed user friendly so that the user can work the System from the tips given in the application itself. Useful tip and guidance is given inside the application itself to help the user. Users have to be made aware that what can be achieved with the new System and how it increases the performance of the System . The user of the System should be given a general idea of the System before he uses the System .

**2.4. SYSTEM MAINTENANCE:**

Maintenance is the enigma of System development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a System is successfully implemented, it should be maintained in a proper manner System maintenance is an important aspect in the software development life cycle. The need for System maintenance is for it to make adaptable to the changes in the System environment. Software maintenance is of course, far more than "Finding Mistakes". Maintenance may be defined by describing four activities that are undertaken after a program is released for use.