Testing is a means of making sure that the product meets the needs of the customer. Software Testing is the process of exercising the software product in pre-defined ways to check if the behavior is the same as expected behavior. The purpose of testing is NOT to prove that the product has no defects [3]. The purpose of software testing is to find defects in software product. The main objectives of testing are to provide quality products to customers.

Defects are like pests, testing is like designing the right pesticides to catch and kill the pests. The test cases that are written are like pesticides.

The purpose of testing a system is to discover faults that cause the system to fail rather than proving the code correctness, which is often an impossible task. In software testing, one is often interested in judging how well a series of test inputs tests a piece of code. A good testing means uncovering as many faults as possible with a potent set of tests. Thus, a test series that has the potential to uncover many faults is better than one that can only uncover a few.

The fundamental principles of the testing are as follows.

i) The goal of testing is to find defects before customer find them out.

ii) Program testing can show the presence of defects, never their absence.

iii) Testing applies all through the software life cycle.

iv) Understand the reason behind the test.

v) Verify the tests first.

vi) Tests develop immunity and have to be revised constantly.

vii) Defects come in clusters or convoys, and testing should focus on these convoys.

viii) Testing encompasses defects prevention.

ix) Intelligent and well planned automation is a key to realizing the benefits of testing.

x) Testing requires talented people who believe in themselves and work in teams.
2.1 SIGNIFICANCE OF TESTING

Almost everything we use today has an element of software in it. Today, in a typical workplace (e.g. Home), just about everyone uses a computer and software. Software today is as common as electricity was in the early part of the last century. Almost every gadget and device we have at home and at work is embedded with software. Mobile phones, televisions, wrist watches etc. all have embedded software. Now, software is being used in some missions were failure is not acceptable. There is no way where one can suggest a solution of “please shutdown and reboot the system” for a software that is in someone’s pacemaker. Almost every service we have taken has software. Banks, air traffic controls, cars are all powered by software these simply cannot afford to fail. These systems have to run reliably, predictably, all the time and every time. To make the system reliable, predictable and to run all the time and every time, software has to be tested before it is delivered.

2.2 CLASSIFICATION OF TESTING

There are different types of testing depends on the test closer to code and user.

2.2.1 Black Box Testing

Black box testing is done without the knowledge of the internals of the system under test. Black box testing is done from customer’s view point. It involves looking at the specifications and does not require examining the code of a program. The test engineers engaged in black box testing only knows the sets of input and expected output and is unaware of how those inputs are transformed into output by software. Black box testing requires functional knowledge of the product to be tested [1, 9].

Black box testing helps in the overall functionality verification of the system under test. It is done based on requirements. The requirements may be stated requirement as well as implied requirements. Black box testing handles valid and invalid inputs. Black box testing attempts to find errors of the following types i.e.

Interface errors; Performance errors; Errors in the data structures or external database access; Incorrect or missing functions; Initialization or termination errors [36].
The black box testing includes various techniques such as Positive or negative testing; Requirement based testing; Boundary value analysis; Domain testing; Equivalence Partitioning [3]

2.2.2 White Box Testing

White box testing is a way of testing the external functionality of the code by examining and testing the program code that realize the external functionality. This is also known as clear box, or glass box or open box testing. It is also called program based testing [38]. White box or logic-driven testing permits you to examine the internal structure of the program. This strategy derives test data from an examination of the program’s logic [37]. White box testing takes into account the program code, code structure, and internal design flow [3]. Its Classification is shown in figure 2.1.

2.2.2.1 Static Testing

This testing requires source code to check errors. It is done by human or with the help of special tools. It never requires the execution of code on computers but involves the tester to go through the code to find whether source code works according to the requirement.
2.2.2.2 Structural Testing

Unlike the static testing, structural testing runs the test cases on the software product that has to be tested with the help of computer. It considers the code, the internal design, code structure, data structure of the software product. There are following types of structural testing [34].

*Statement Coverage Testing*

Every statement of the software/program under test has to be executed at least once during testing. The main drawback of statement testing is that even if one achieves a very high level of statement coverage, it does not reflect that program is error free.

*Branch Coverage Testing*

Every branch and every entry point of the software/program under test has to be executed at least once during testing. In this testing all control transfer are executed [39].

This is much stronger criteria as compared to statement coverage as each statement is covered with the execution of every branch of the program. However some errors can only be detected if the statements and branches are executed in a certain order [34].

*Path Coverage Testing*

It is stronger criteria than Statement and Branch Coverage Testing. In Path Coverage Testing, every possible path is executed. This tries to find out the percentage of code coverage to more extent and hence increase the chances of error detection. Path Coverage Testing becomes impractical if there is loop in the program.

*Data Flow Testing*

Data Flow Testing is based on the flow of control of a program. Data Flow analysis focuses on the interactions between variable definitions (defs) and references (uses) in a program. Variable uses can be split into 'c-uses' and 'p-uses' according to whether the variable use occurs in a computation or a predicate. Defs and c-uses are associated with nodes, but p-uses are associated with edges. The purpose of the data flow analysis is to determine the defs of every variable in the program and the uses that might be affected by these defs, i.e. the def-use associations. Such data flow relationships can be represented by the following two sets: dcu(i), the set of all
variable defs for which there are def-clear paths to their uses at node i; and dpu(i,j),
the set of all variable defs for which there are def-clear paths to their p-uses at edge
(i,j) [14].

2.3 TESTING PROBLEMS

To test the software, test cases are written. In order to find out how a test case
is valid, one does not have a definite mechanism. One basically depends on the testers
understanding of the requirement. In this process, one has lot of human error and his
basic skill level taken into consideration. This leads to the inclusion of bugs in the
system after testing also. To overcome this, it is essential to Automate Test Data
Generation.

Automated test data generation reduces an effort of software developers for
creating test cases with a goal to minimize the amount of manual work involved in
text execution and gain higher coverage with minimum cost and time.

2.4 TEST AUTOMATION

Developing software to test software is called automation. There are two way
to create test cases that are used to test software – Automated, Manual.

Consider a program that is supposed to accept a six character code and ensure
that first character is numeric and rests of the characters are alphanumeric. If we
check each and every combination as input, then it would be $10 \times (62^5)$ combination,
if each combination takes 10 seconds to run , then testing all these valid combinations
will take 2905 years approximately. However we can choose to execute only a subset
of tests that can uncover the maximum numbers of errors. Some techniques such as
equivalence partitioning, boundary value analysis, code path analysis and so on,
which helps in identifying subsets of test cases that have higher likelihood of
uncovering errors. But using any technique we have to choose test cases manually
then run them to test, this manual work requires a lot of time, cost and human skill. To
overcome the problem of shortage of skilled persons, and to reduce the time and cost
of testing, software testing automation is required. If the testing process could be
automated, the cost of developing software could be reduced significantly. Test
automation can help address several problems.

Automation saves times as software can execute test cases faster than human
do. This helps in running the tests overnight or unattended saving the elapsed time for
testing, therefore enabling the product to be released frequently. The time thus saved can be used effectively for test engineers to develop additional test cases. Therefore the coverage of testing is improved. Advantages of automated testing than manual testing are

i) Test automation can free the test engineers from mundane task and make them focus on more creative tasks.

ii) Automated tests are more reliable.

iii) Automation helps in immediate testing

iv) Automation can protect an organization against attrition of test engineers.

v) Test automation opens up opportunities for better utilization of global resources.

vi) Certain types of testing cannot be executed without automation – e.g. tests cases for reliability testing, stress testing.

2.4.1 Architecture of Test Automation

Design and architecture is an important aspect of automation. The architecture of test automation is shown in figure 2.2. Architecture for test automation involves two major heads:

![Diagram of Test Automation Architecture]

Figure 2.2 Test Automation Architecture
i) A test infrastructure that covers a test case database

ii) A defect database or defect repository

There is no hard and fast rule on when automation should start and when it should end. The work on automation can go simultaneously with product development. It can overlap with multiple release of the product. Product and automation go parallel in the same direction with similar expectation.

2.4.2 Criteria for Selection of Testing Tool

Selecting a test tool for automation is an important aspect of test automation for several reasons as given below:

i) Free tools are not well supported and get phase out soon.

ii) Developing in house tools takes time.

iii) Test tools sold by vendors are expensive.

iv) Test tools require strong training.

v) Test tools generally not meet all requirements for automation.

vi) Not all test tools run on all platform.