

TESTING

Coding is undertaken once the design phase is complete and the design documents have been successfully reviewed.

In the coding phase, every module specified in the design document is coded and unit tested. During unit testing, each module is tested in isolation from other modules. That is, a module is tested independently as and when its coding is complete.

After all the modules of a system have been coded and unit tested, the integration and system testing phase is undertaken.

Integration and testing of modules is carried out according to an integration plan. The integration plan, according to which different modules are integrated together, usually envisages integration of modules through a number of steps. During each integration step, a number of modules are added to the partially integrated system and the resultant system is tested.

The full product takes shape only after all the modules have been integrated together. System testing is conducted on the full product. During system testing, the product is tested against its requirements as recorded in the SRS document.

Usually, testing of a professional software is carried out using a large number of test cases. In a typical development organisation, at any time, the maximum number of software engineers can be found to be engaged in testing activities. It is not very surprising then that in the software industry there is always a large demand for software test engineers.

SOFTWARE DOCUMENTATION

When a software is developed, in addition to the executable files and the source code, several kinds of documents such as users' manual, software requirements specification (SRS) document, design document, test document, installation manual, etc., are developed as part of the software engineering process. All these documents are considered a vital part of any good software development practice.

Different types of software documents can broadly be classified into the following:

Internal Documentation

Internal documentation is the code comprehension features provided in the source code itself. Internal documentation can be provided in the code in several forms. The important types of internal documentation are the following:

- Comments embedded in the source code.
- Use of meaningful variable names.
- Module and function headers.
- Code indentation.
- Code structuring (i.e., code decomposed into modules and functions).
- Use of user-defined data types.

Careful experiments suggest that out of all types of internal documentation, meaningful variable names is most useful while trying to understand a piece of code.

External Documentation

External documentation is provided through various types of supporting documents such as users' manual, software requirements specification document, design document, test document, etc. A systematic software development style ensures that all these documents are of good quality and are produced in an orderly fashion.

SOFTWARE TESTING

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. The aim of program testing is to help identify all defects in a program. However, in practice, even after satisfactory completion of the testing phase, it is not possible to guarantee that a program is error free. This is because the input data domain of most programs is very large, and it is not practical to test the program exhaustively with respect to each value that the input can assume.

Testing a program involves executing the program with a set of test inputs and observing if the program behaves as expected. If the program fails to behave as expected, then the input data and the conditions under which it fails are noted for later debugging and error correction.

Software testing comprises of Validation and Verification.

Software Validation

Validation is process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which attempts all that user needs from this software ?".
- Validation emphasizes on user requirements.

Software Verification

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by firmly following all design specifications ?"
- Verifications concentrates on the design and system specifications.

Target of the test are -

- **Errors** - These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, is considered as an error.
- **Fault** - When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause system to fail.
- **Failure** - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

TESTING ACTIVITIES

Testing involves performing the following main activities:

1.Test suite design: A **test case** is a triplet [I, S, R], where I is the data input to the program under test, S is the state of the program at which the data is to be input, and R is the result expected to be produced by the program. The state of a program is also called its execution mode.

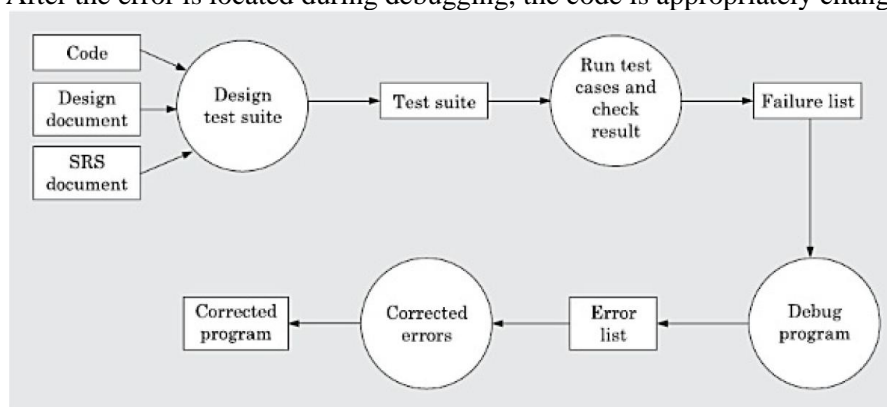
In simple words, we can say that a test case is a set of test inputs, the mode in which the input is to be applied, and the results that are expected during and after the execution of the test case.

A **test suite** is the set of all test that have been designed by a tester to test a given program. The set of test cases using which a program is to be tested is designed possibly using several test case design techniques.

2.Running test cases and checking the results to detect failures: Each test case is run and the results are compared with the expected results. A mismatch between the actual result and expected results indicates a failure. The test cases for which the system fails are noted down for later debugging.

3.Locate error: In this activity, the failure symptoms are analysed to locate the errors. For each failure observed during the previous activity, the statements that are in error are identified.

4.Error correction: After the error is located during debugging, the code is appropriately changed to correct the error.



MANUAL VS AUTOMATED TESTING

Testing can either be done manually or using an automated testing tool:

Manual - This testing is performed without taking help of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests and reports the result to the manager.

Manual testing is time and resource consuming. The tester needs to confirm whether or not right test cases are used. Major portion of testing involves manual testing.

Automated - This testing is a testing procedure done with aid of automated testing tools. The limitations with manual testing can be overcome using automated test tools.

A test needs to check if a webpage can be opened in Internet Explorer. This can be easily done with manual testing. But to check if the web-server can take the load of 1 million users, it is quite impossible to test manually.

There are software and hardware tools which help tester in conducting load testing, stress testing, regression testing.

TESTING APPROACHES

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

When functionality is being tested without taking the actual implementation in concern it is known as black-box testing. The other side is known as white-box testing where not only functionality is tested but the way it is implemented is also analyzed.

Exhaustive tests are the best-desired method for a perfect testing. Every single possible value in the range of the input and output values is tested. It is not possible to test each and every value in real world scenario if the range of values is large.

BLACK-BOX TESTING

It is carried out to test functionality of the program. It is also called ‘Behavioral’ testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested ‘ok’, and problematic otherwise.



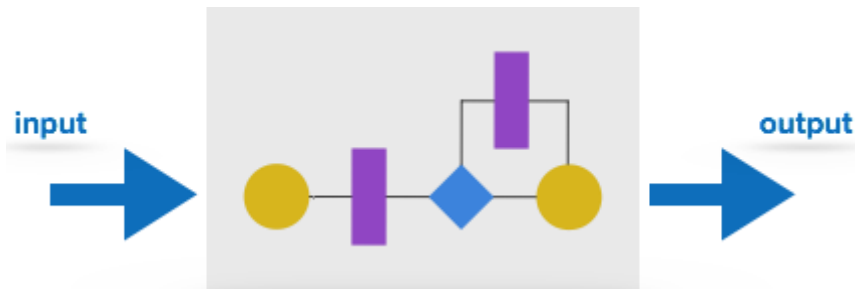
In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

Black-box testing techniques:

- **Equivalence class** - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- **Boundary values** - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
- **Cause-effect graphing** - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- **Pair-wise Testing** - The behavior of software depends on multiple parameters. In pairwise testing, the multiple parameters are tested pair-wise for their different values.
- **State-based testing** - The system changes state on provision of input. These systems are tested based on their states and input.

WHITE-BOX TESTING

It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as ‘Structural’ testing.



In this testing method, the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

The below are some White-box testing techniques:

- **Control-flow testing** - The purpose of the control-flow testing to set up test cases which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- **Data-flow testing** - This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

TESTING LEVELS

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified.

Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Software is tested on various levels –

UNIT TESTING

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

INTEGRATION TESTING

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updation, etc.

The objective of integration testing is to check whether the different modules of a program interface with each other properly. In other words, the objective is to detect the errors at the module interfaces (call parameters). For example, it is checked that no parameter mismatch occurs when one module invokes the functionality of another module. Thus, the primary objective of integration testing is to test the module interfaces, i.e., there are no errors in parameter passing, when one module invokes the functionality of another module.

During integration testing, different modules of a system are integrated in a planned manner using an integration plan. The integration plan specifies the steps and the order in which modules are combined to realize the full system. After each integration step, the partially integrated system is tested.

SYSTEM TESTING

After all the units of a program have been integrated together and tested, system testing is taken up. System tests are designed to validate a fully developed system to assure that it meets its requirements.

The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against the requirement.
- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.
- **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

ACCEPTANCE TESTING

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

- **Alpha testing** - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.
- **Beta testing** - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

REGRESSION TESTING

Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code. This is known as regression testing.