VARIOUS MAJOR TESTING STRATEGIES

To perform testing in a planned and systematic manner, software testing strategy is developed. A testing strategy is used to identify the levels of testing which are to be applied along with the methods, techniques, and tools to be used during testing. This strategy also decides test cases, test specifications, test case decisions, and puts them together for execution.

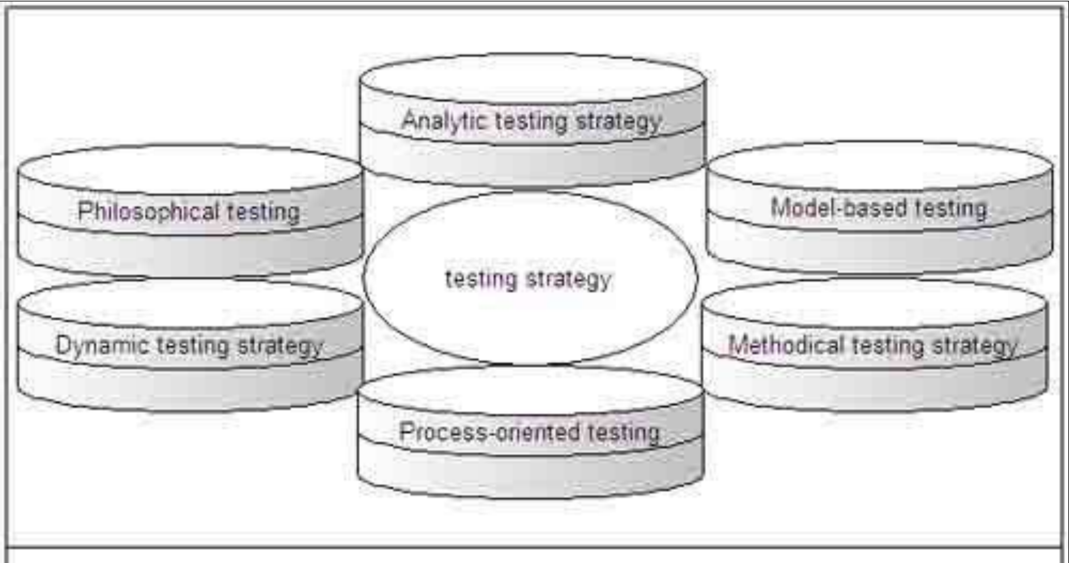
A number of software testing strategies are developed in the testing process. All these strategies provide the tester a template, which is used for testing.

Generally, all testing strategies have following characteristics:

1. Testing proceeds in an outward manner. It starts from testing the individual units, progresses to integrating these units, and finally, moves to system testing.
2. Testing techniques used during different phases of software development are different.
3. Testing is conducted by the software developer and by an ITG.
4. Testing and debugging should not be used synonymously. However, any testing strategy must accommodate debugging with itself.

TYPES OF SOFTWARE TESTING STRATEGIES

There are different types of software testing strategies, which are selected by the testers depending upon the nature and size of the software.



Types of Software Testing Strategy

The commonly used software testing strategies are:

1. **Analytic testing strategy:**
This uses formal and informal techniques to access and prioritize risks that arise during software testing. It takes a complete overview of requirements, design, and implementation of objects to determine the motive of testing. In addition, it gathers complete information about the software, targets to be achieved, and the data required for testing the software.

2. **Model-based testing strategy:**
This strategy tests the functionality of the software according to the real world scenario (like software functioning in an organization). It recognizes the domain of data and selects suitable test cases according to the probability of errors in that domain.

3. **Methodical testing strategy:**
It tests the functions and status of software according to the checklist, which is based on user requirements. This strategy is also used to test the functionality, reliability, usability, and performance of the software.

4. **Process-oriented testing strategy:**
It tests the software according to already existing standards such as the IEEE standards. In addition, it checks the functionality of the software by using automated testing tools.

5. **Dynamic testing strategy:**
This tests the software after having a collective decision of the testing team. Along with testing, this strategy provides information about the software such as test cases used for testing the errors present in it.

6. **Philosophical testing strategy:**
It tests the software assuming that any component of the software can stop functioning anytime. It takes help from software developers, users and systems analysts to test the software.

The output produced by the software testing strategy includes a detailed document, which indicates the entire test plan including all test cases used during the testing phase. A testing strategy also specifies a list of testing issues that need to be resolved.

An efficient software testing strategy includes two types of tests, namely, low-level tests and high-level tests. Low-level tests ensure correct implementation of small part of the source code and high-level tests ensure that major software functions are validated according to user requirements. A testing strategy sets certain milestones for the software such as final date for completion of testing and the date of delivering the software. These milestones are important when there is limited time to meet the deadline.

There are certain issues that need to be addressed for successful implementation of software testing strategy :

1. In addition to detecting errors, a good testing strategy should also assess portability and usability of the software.
2. It should use quantifiable manner to specify software requirements such as outputs expected from software, test effectiveness, and mean time to failure which should be clearly stated in the test plan.
3. It should improve testing method continuously to make it more effective.
4. Test plans that support rapid cycle testing should be developed. The feedback from rapid cycle testing can be used to control the corresponding strategies.
5. It should develop robust software, which is able to test itself using debugging techniques.
6. It should conduct formal technical reviews to evaluate the test cases and test strategy. The formal technical reviews can detect errors and inconsistencies present in the testing process.