Test Strategy

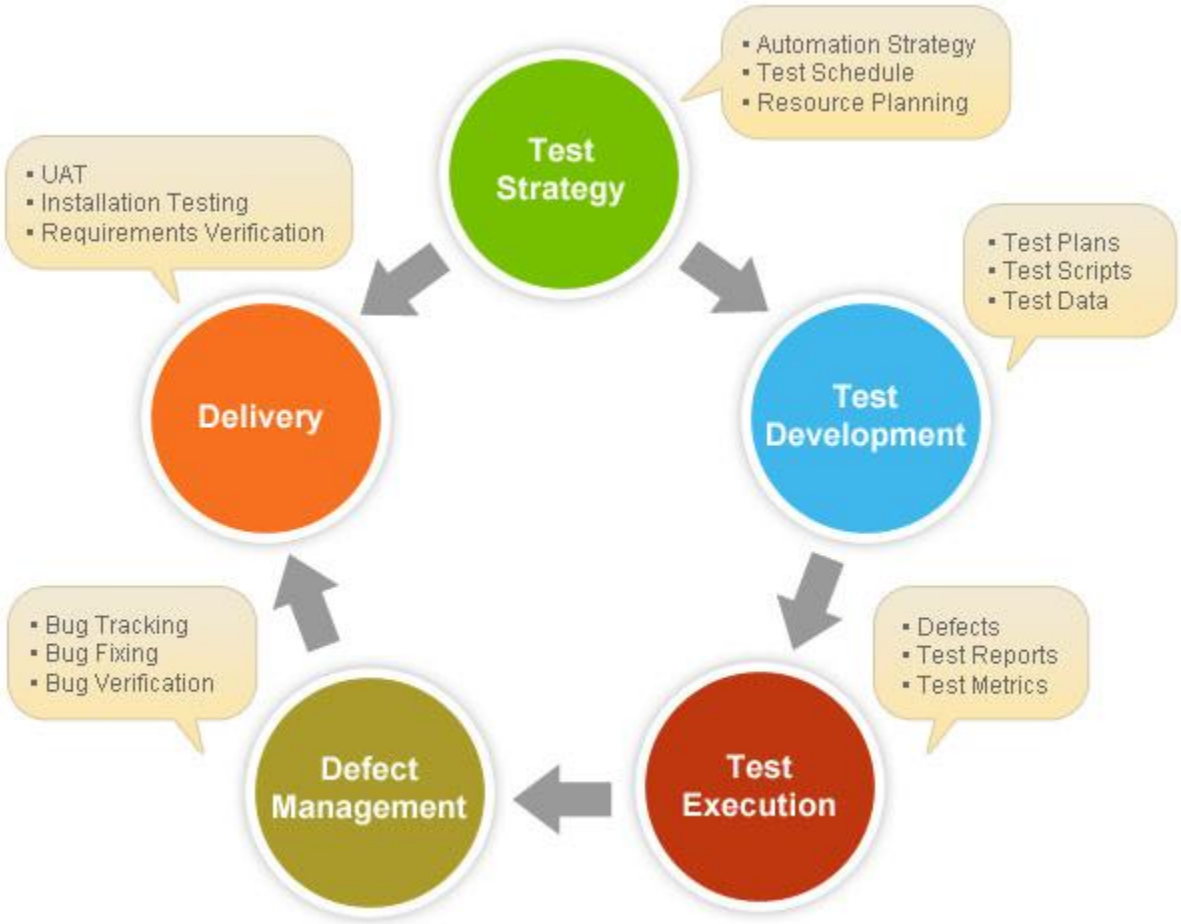
Test Strategy is summarize or outline which describes the approach of software development cycle.

Or

The test levels to be performed in testing and the description of testing activities within those test levels are known as test strategy.

It is created to inform project managers, testers and developers about some key points of testing process. Strategy also includes testing objective, methods to testing new functions, time and resource what will be used in project and testing environment.

Test strategies describe how to mitigate the product risk at test level, which of types of test are to be performed in testing and which entry and exit criteria apply. Different Testing strategies may be adopted depending on the type of system to be test and the development process used.



In this document we are going to discuss testing strategies on two different types of software testing.

1. Conventional Software
2. Object Oriented Software(OO software)

Conventional Software Testing: It is traditional approach. It takes place mostly when water fall life cycle is used for development. Conventional software Testing focuses more on decomposition. Conventional software testing always takes place during the test phase of life cycle, which usually follows the development phase and proceeds the implementation phase. During the conventional testing phase, Three types of testing will be conducted

1. System Testing
2. Integration Testing
3. Unit Testing

Object Oriented Software Testing: Using Object Oriented design or design along with Agile development methodology leads to Object Oriented Software Testing. Object Oriented testing is done having emphasis on Composition. This means design is created piece by piece and then composed together to complete full system. Three Conventional testing types also used in Object Oriented Software testing but these three are not clearly visible in Object oriented Design.

System Testing takes much of same approach as conventional Software Testing. Unit testing under object oriented testing is similar as conventional testing except definition of the unit used (currently Unit testing are classes and method in Object oriented Software testing).

Strategy for Conventional Software Testing:

a. **Unit testing:** we know that testing begins with unit testing. It aims to testing functionality of each component or unit of software and ensures that it works properly as a unit. Some typical units are:
b. **Integration Testing:** In further process of testing, units must be assembled or group to form a software package. So Integration testing takes place and focuses on problems of composition.
c. **Validation Testing:** provide assurance that the software validation criterion meets all functional, behavioral and performance requirement.
d. **System Testing:** verifies that all system components mesh properly and overall system function and performance has been achieved. It includes recovery testing, stress testing, security testing, performance testing.

Testing Strategy for Object Oriented software testing

a. **Unit Testing :** components being used are classes not modules
b. **Integration Testing:** as classes are integrated in software regression tests are run to uncover errors between objects
c. **System Testing:** the system as a whole is tested to uncover requirement errors.

There are some different strategies between Unit testing and Integration testing of both types of software testing (Conventional software testing and Object oriented Software testing)

Unit Testing (Conventional Software Testing)

- Module interfaces are tested for proper information flow.
- Local data are examined to ensure that integrity is maintained.
- Boundary conditions are tested.
- Basis (independent) path are tested.
- All error handling paths should be tested.
- Drivers and/or stubs need to be developed to test incomplete software.

Driver: A simple main program that accepts test case data, passes such data to the component being tested, and prints the returned results

Stubs :

1. **Serve to replace modules that are subordinate to (called by) the component to be tested**
2. **It uses the module's exact interface, may do minimal data manipulation, provides verification of entry, and returns control to the module undergoing testing**

Unit Testing (Object Oriented testing)

- smallest testable unit is the encapsulated class or object
- similar to system testing of conventional software
- do not test operations in isolation from one another
- driven by class operations and state behavior, not algorithmic detail and data flow across module interface

Integration Testing (Conventional Software Testing)

Top-down integration testing

1. Main control module used as a test driver and stubs are substitutes for components directly subordinate to it.
2. Subordinate stubs are replaced one at a time with real components (following the depth-first or breadth-first approach).
3. Tests are conducted as each component is integrated.
4. On completion of each set of tests and other stub is replaced with a real component.
5. Regression testing may be used to ensure that new errors not introduced.

Bottom-up integration testing

1. Low level components are combined into clusters that perform a specific software function.
2. A driver (control program) is written to coordinate test case input and output.
3. The cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.

Regression Testing: used to check for defects propagated to other modules by changes made to existing program

1. Representative sample of existing test cases is used to exercise all software functions.
2. Additional test cases focusing software functions likely to be affected by the change.
3. Tests cases that focus on the changed software components.

Smoke Testing

1. Software components already translated into code are integrated into a build.
2. A series of tests designed to expose errors that will keep the build from performing its functions are created.
3. The build is integrated with the other builds and the entire product is smoke tested daily (either top-down or bottom integration may be used).

Integration Testing (Object-Oriented Software Testing)

- focuses on groups of classes that collaborate or communicate in some manner integration of operations one at a time into classes is often meaningless
- **Thread-based testing**: testing all classes required to respond to one system input or event
- **use-based testing**: begins by testing independent classes (classes that use very few server classes) first and the dependent classes that make use of them
- **cluster testing**: groups of collaborating classes are tested for interaction errors regression testing is important as each thread, cluster, or subsystem is added to the system

Common Sections of test strategy document:

1. **Scope and overview:** describe project overview with information like who will use this document, who will review and who approve this. Define testing activities and phases to be carried out with timeline with respect to overall project timeline defined in test plan document.
2. **Test Approach:** Define testing process, how many level of testing used, what are roles and responsibilities of team member in this section. Besides this needs to define types of testing (unit testing, integration testing, system testing etc) with details like when to start, who is test owner, test approach and if applicable define automation strategy and tool.
3. **Test Environment:** setup the outline information like number of environments and required setup for each environment to different team like a environment for functional test team, other one for automation testing team. Should be defined number of users supported to environment, access roles of each user and software, hardware requirements
4. **Testing tools:** Define the tools which are required for test execution like test management and automation tools. Describe strategy for load, performance testing and list the tools which are required for these testing.
5. **Risk Analysis:** List all the risk what you visualize and provide a proper plan to mitigate the risks.