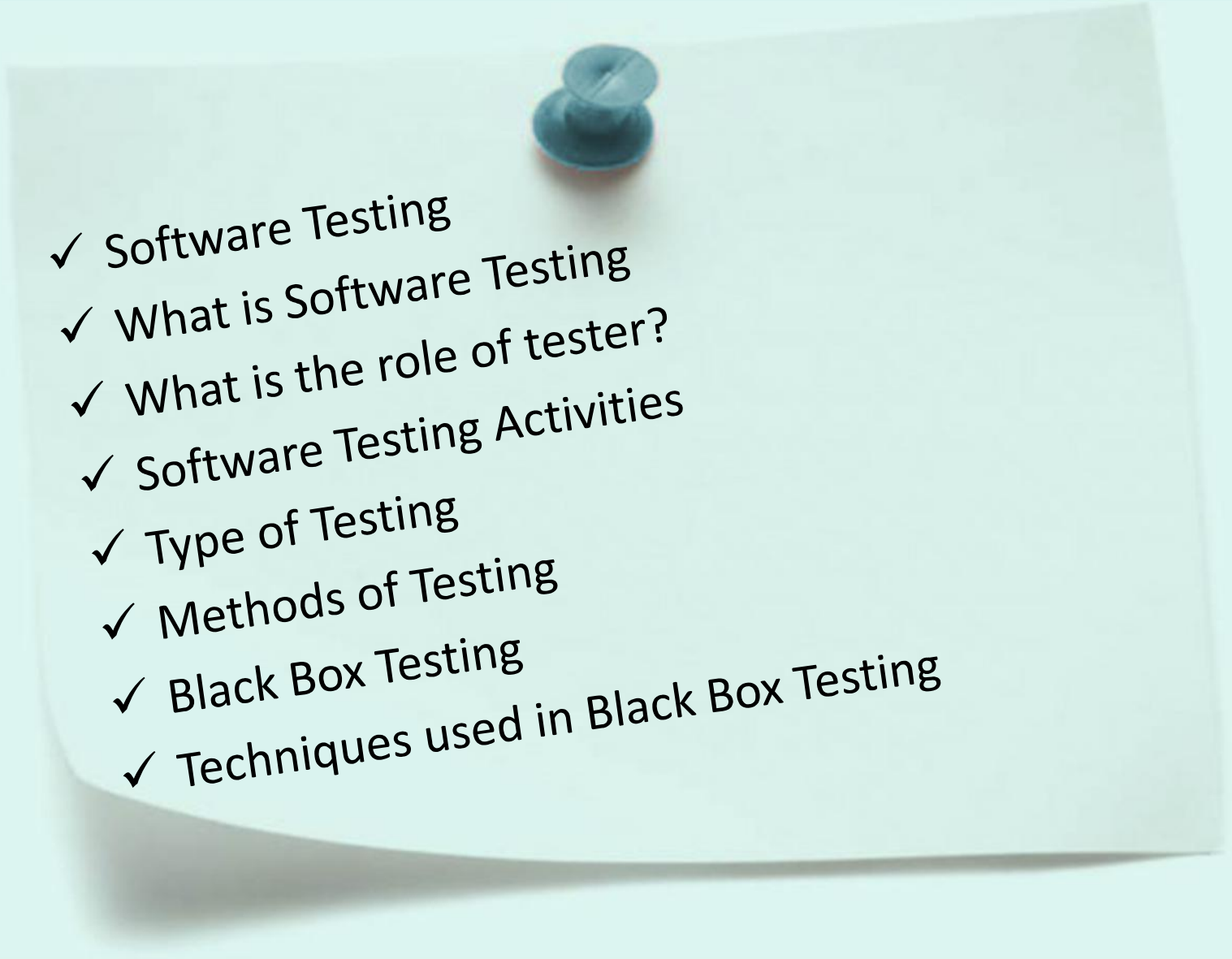
Software Engineering -1

- **Engr. M. Fahad Khan**
- Lecturer, Software Engineering Department
- University of Engineering & Technology, Taxila

Topics covered



- ✓ Software Testing
- ✓ What is Software Testing
- ✓ What is the role of tester?
- ✓ Software Testing Activities
- ✓ Type of Testing
- ✓ Methods of Testing
- ✓ Black Box Testing
- ✓ Techniques used in Black Box Testing

Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software Testing

Software testing can be stated as the process of validating and verifying that a software program/application/product:

✓meets the requirements that guided its design and development
✓works as expected
✓can be implemented with the same characteristics.

What is Software Testing

Several definitions:

“Testing is the process of establishing confidence that a program or system does what it is supposed to.” by Hetzel 1973

“Testing is the process of executing a program or system with the intent of finding errors.” by Myers 1979

Testing Objectives

The Major Objectives of Software Testing:

1 Identify the magnitude and sources of development risk reducible by testing.

2 Perform testing to reduce identified risks.

3 Know when testing is completed.

4 Manage testing as a standard project within the development project.

Major Goals

uncover the errors (defects) in the software, including errors in:

- requirements from requirement analysis
- design documented in design specifications
- coding (implementation)
- system resources and system environment
- hardware problems and their interfaces to software

Why we go for testing?

Well, while making food, its ok to have something extra, people might understand and eat the things you made and may well appreciate your work. But this isn't the case with Software Project Development. If you fail to deliver a reliable, good and problem free software solution, you fail in your project and probably you may loose your client. This can get even worse!

So in order to make it sure, that you provide your client a proper software solution, you go for TESTING. You check out if there is any problem, any error in the system, which can make software unusable by the client. You make software testers test the system and help in finding out the bugs in the system to fix them on time. You find out the problems and fix them and again try to find out all the potential problems.

What is the role of tester?

A tester is a person who tries to find out all possible errors/bugs in the system with the help of various inputs to it. A tester plays an important part in finding out the problems with system and helps in improving its quality.

If you could find all the bugs and fix them all, your system becomes more and more reliable.

A tester has to understand the limits, which can make the system break and work abruptly. The more number of VALID BUGS tester finds out, the better tester he/she is!

What is the role of tester?

Software Errors, Defect & Fault

What is a software error?

Definition #1:

"A mismatch between the program and its specification is an error in the program if and only if the specification exists and is correct."

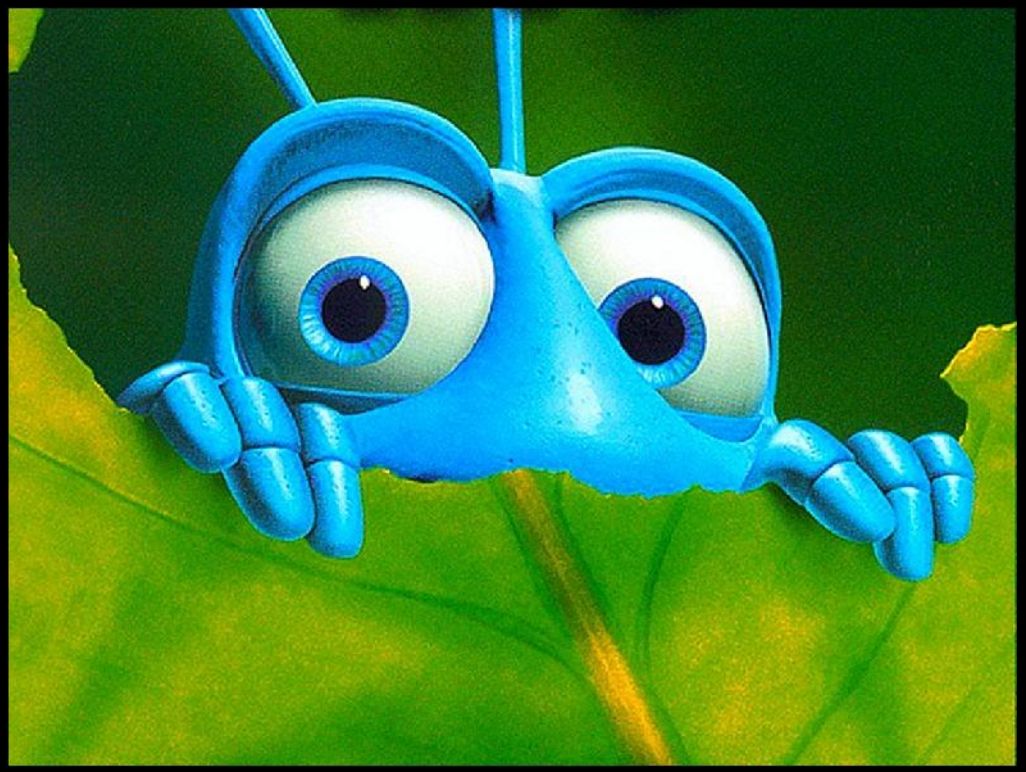
Definition #2:

"A quality problem that is discovered by software engineers or others before the software is released to end user.

software Defect & Software Fault?

A quality problem that is discovered after the software has been released to end users

Software Errors, Defect & Fault



Software Testing Activities

- Test Planing
  Define a software test plan by specifying:
  - a test schedule for a test process and its activities, as well as assignments
  - test requirements and items
  - test strategy

- Test Design and Specification
  - Conduct software design based well-defined test generation methods.
  - Specify test cases to achieve a targeted test coverage.

- Test Set up:
  - Testing Tools and Environment Set-up
  - Test Suite Set-up

- Test Operation and Execution
  - Run test cases manually or automatically

Software Testing Activities

- Test Result Analysis and Reporting

Report software testing results and conduct test result analysis

- Problem Reporting

Report program errors using a systematic solution.

- Test Management and Measurement

Manage software testing activities, control testing schedule, measure testing complexity and cost

- Test Automation

- Define and develop software test tools
- Adopt and use software test tools
- Write software test scripts and facility

- Test Configuration Management

- Manage and maintain different versions of software test suites, test environment and tools, and documents for various product versions.

Software Testing Activities

Verification and Validation

Software testing is one element of a broader topic that is often referred to as
===> Verification and Validation (V&V)

Verification --> refers to the set of activities that ensure that software correctly implements a specific function.

Validation --> refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

Boehm [BOE81]:

Verification: “Are we building the product right?”

Validation: “Are we building the right product?”

When To Stop Testing

Its difficult to determine exactly when to stop Testing. The following are some of the cases which can help to decide when to reduce/stop Testing:

1)Deadlines(Release/Test Deadlines etc)

2)Test cases completed with certain percentage passed.

3)Test budget depleted.

4)Coverage of code/functionality requirements reaches a specified point.

5)Bug rate falls below a certain level.

6)Beta or Alpha testing period ends.

Software Testing Myths

- We can test a program completely. In other words, we test a program exhaustively.

- We can find all program errors as long as test engineers do a good job.

- We can test a program by trying all possible inputs and states of a program.

- A good test suite must include a great number of test cases.

- Good test cases always are complicated ones.

- Software test automation can replace test engineers to perform good software testing.

- Software testing is simple and easy. Anyone can do it. No training is needed.

Software Testing Limits

- Due to the testing time limit, it is impossible to achieve total confidence.

- We can never be sure the specifications are 100% correct.

- We can never be certain that a testing system (or tool) is correct.

- Test engineers never be sure that they completely understand a software product.

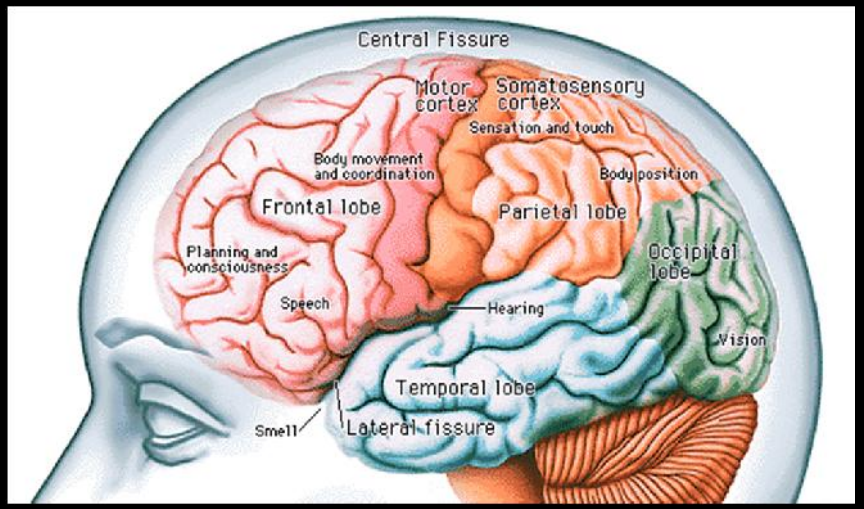
- We never have enough resources to perform software testing.

- We can never be certain that we achieve 100% adequate software testing.

The Mind of a Tester

Kaner, Bach, and Pettichord describe four different kinds of thinking exhibited by a good tester:

1. Technical thinking: the ability to model technology and understand causes and effects
2. Creative thinking: the ability to generate ideas and see possibilities
3. Critical thinking: the ability to evaluate ideas and make inferences
4. Practical thinking: the ability to put ideas into practice

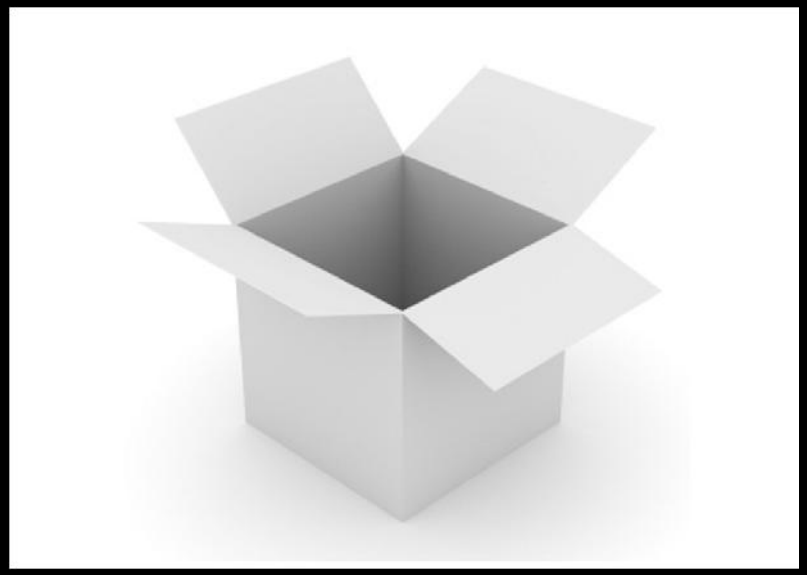


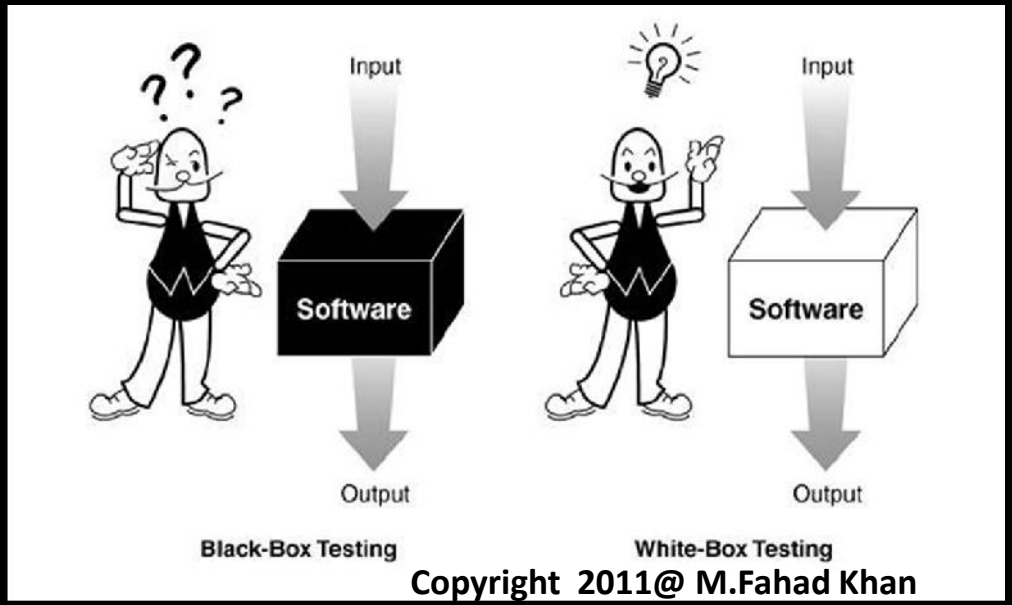
Type of testing

- ✓White Box Testing
- ✓Black Box Testing
- ✓Grey box testing

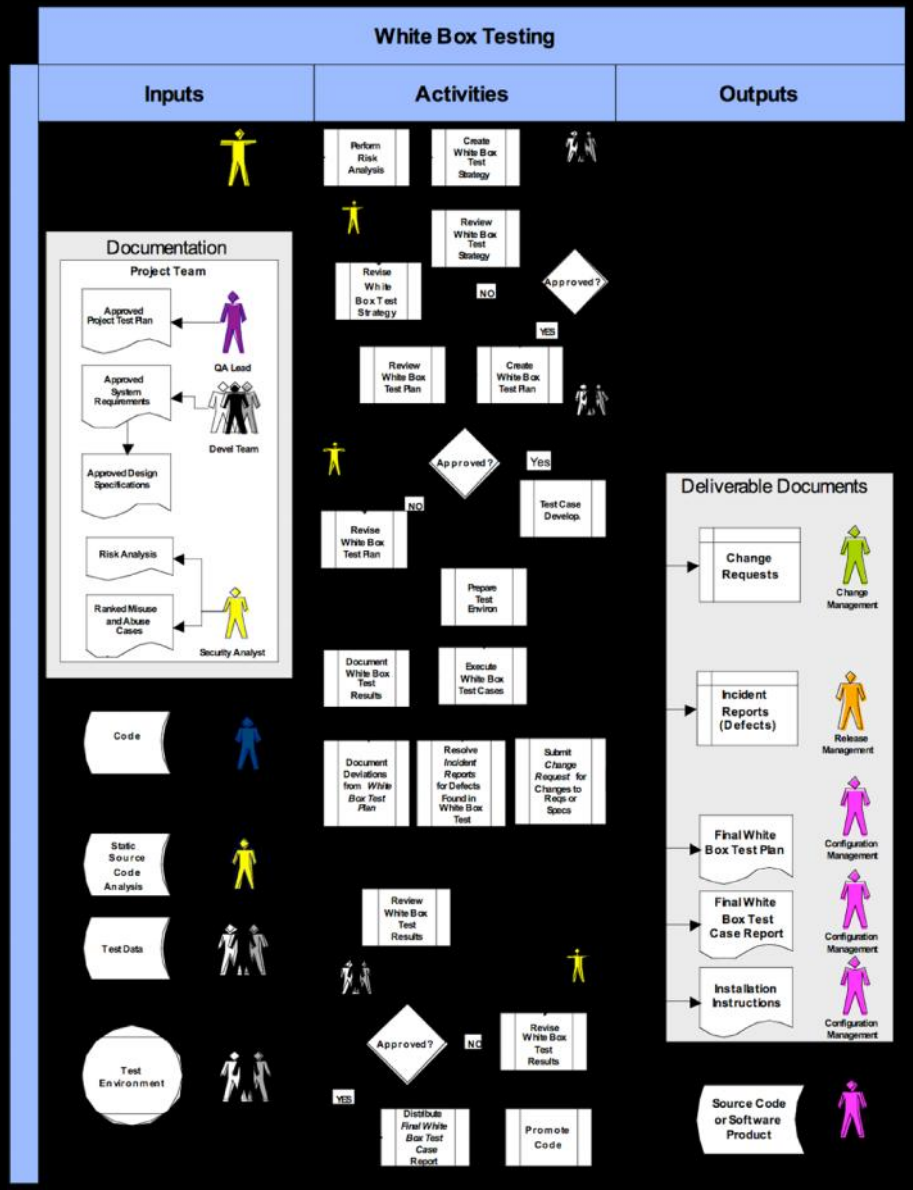
White Box Testing

- **White box testing** is when the tester has access to the internal data structures and algorithms including the code that implement these.
- White box testing is a strategy in which testing is based on the internal paths, structure, and implementation of the software under test. Unlike its complement, black box testing, white box testing generally requires detailed programming skills.





White Box Testing



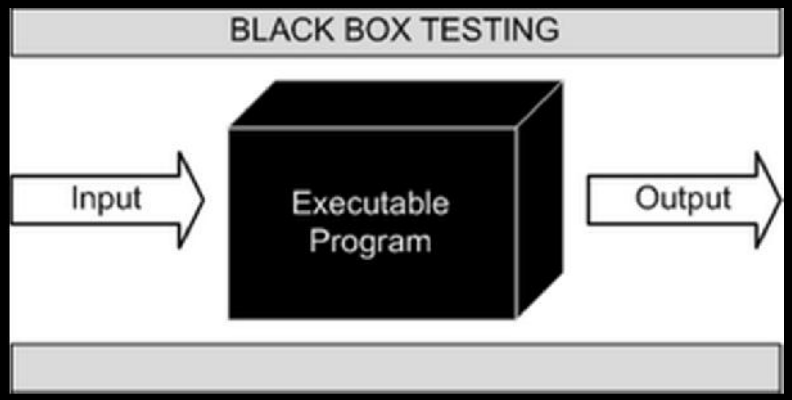
Types of white box testing

- Types of white box testing The following types of white box testing exist: **API** testing (application programming interface) - testing of the application using public and private APIs
- **Code coverage** - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- **Fault injection** methods - improving the coverage of a test by introducing faults to test code paths

Black Box Testing

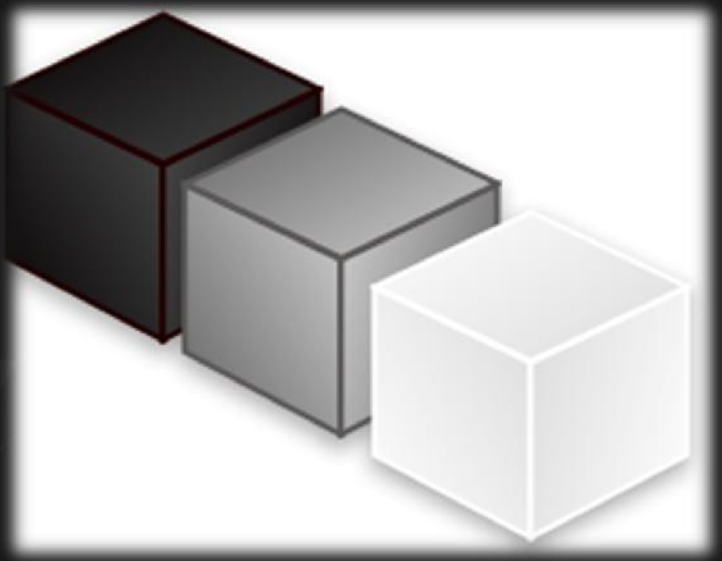
- Black box testing treats the software as a "black box"—without any knowledge of internal implementation.
- Black box testing is a strategy in which testing is based solely on the requirements and specifications. Unlike its complement, white box testing, black box testing requires no knowledge of the internal paths, structure, or implementation of the software under test.
- **Specification-based testing**: Specification-based testing aims to test the functionality of software according to the applicable requirements.Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case.

Black Box Testing

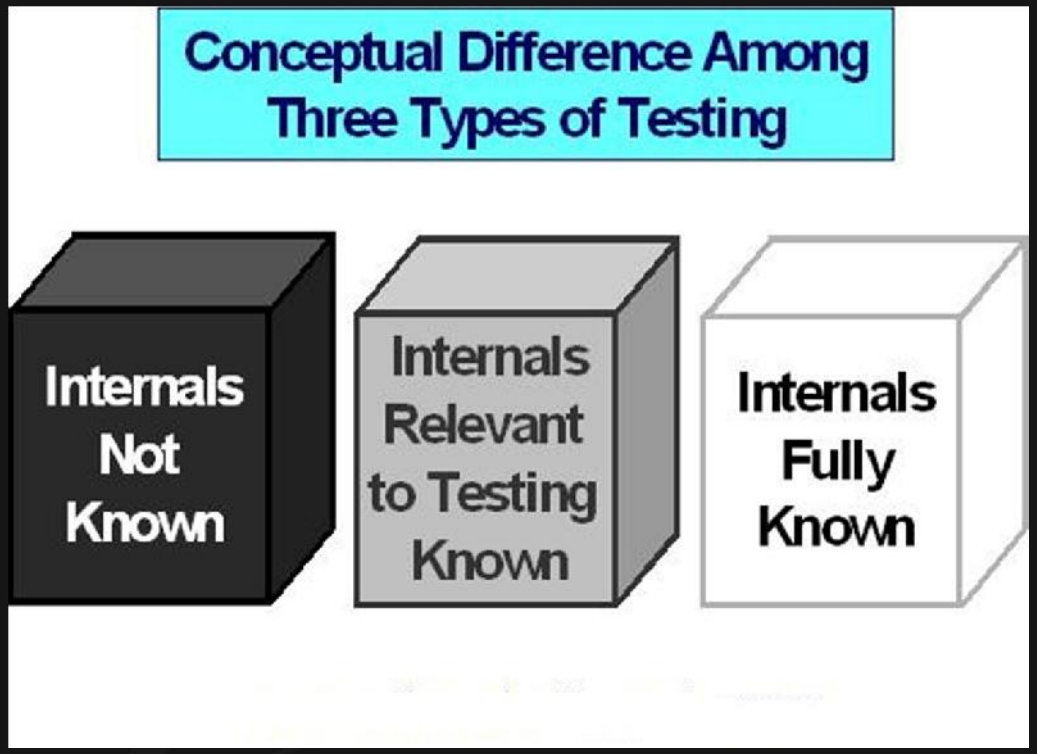


Gray Box Testing

Grey box testing (American spelling: **gray box testing**) involves having knowledge of internal data structures and algorithms for purposes of designing the test cases.



Difference



Testing levels

System testing

System testing tests a completely integrated system to verify that it meets its requirements.

System integration testing

System integration testing verifies that a system is integrated to any external or third-party systems defined in the system requirements.

Testing levels

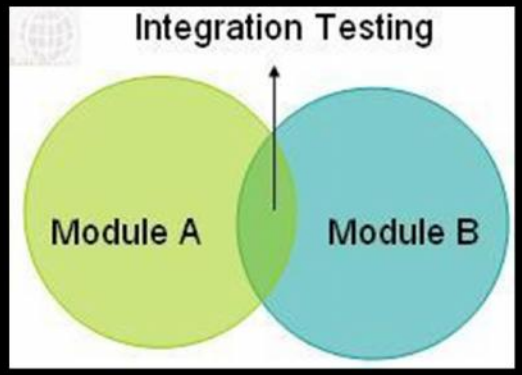
Unit testing

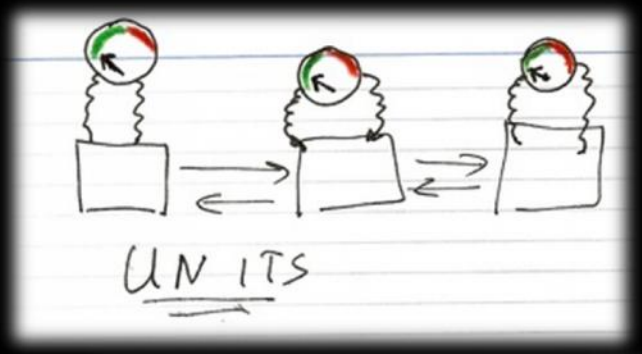
Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localized more quickly and fixed.

Unit testing Vs Integration testing



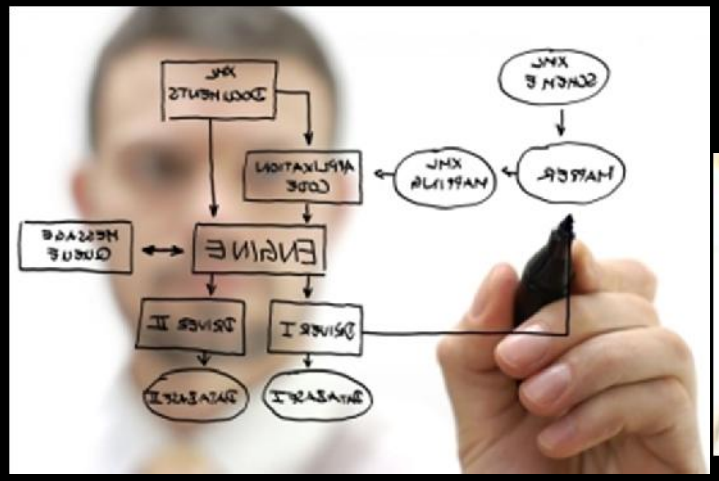


Objectives of testing

Regression testing

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, or old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly stops working as intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the risk of the added features. They can either be complete, for changes added late in the release or deemed to be risky, to very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of low risk.

Regression Testing



Acceptance testing

- **Alpha testing**

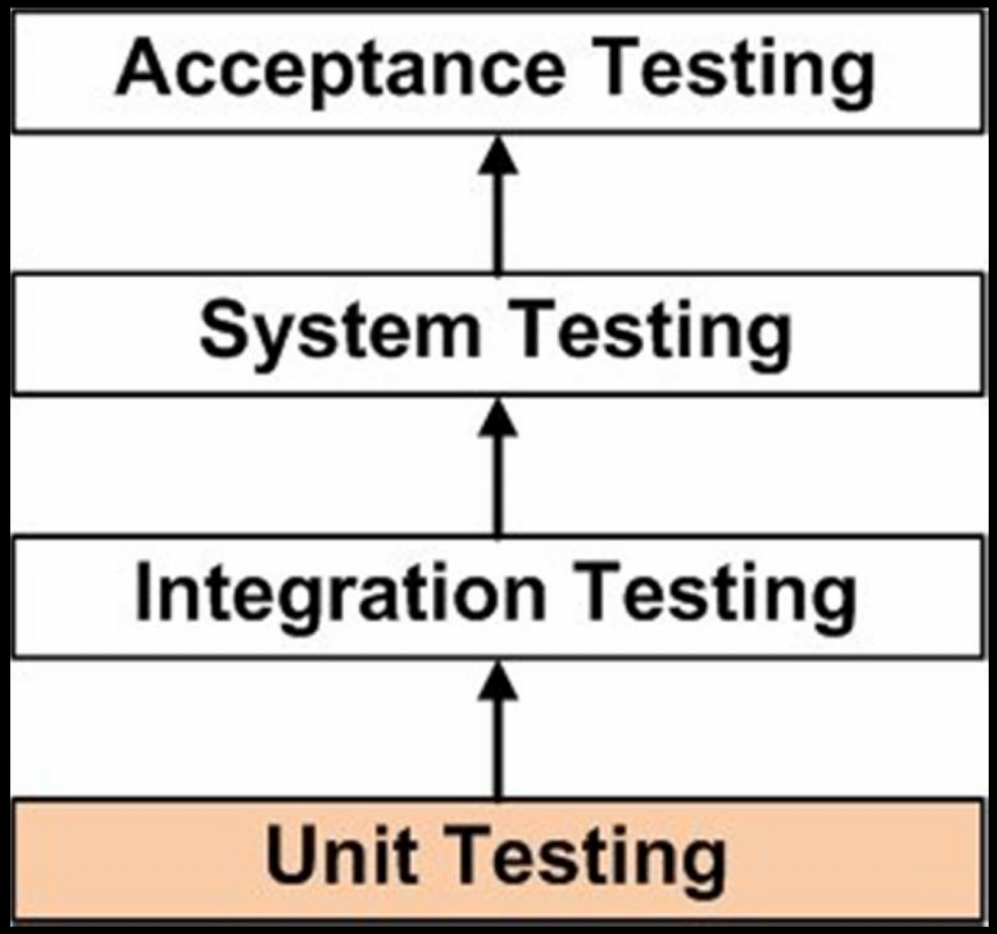
  Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

- **Beta testing**

  Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Alpha testing Vs Beta testing

Testing Types



Software performance testing and load testing

- **Performance testing** is executed to determine how fast a system or sub-system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system, such as scalability, reliability and resource usage. **Load testing** is primarily concerned with testing that can continue to operate under a specific load, whether that be large quantities of data or a large number of users. This is generally referred to as software **scalability.** The related load testing activity of when performed as a non-functional activity is often referred to as *endurance testing*.
- **Volume testing** is a way to test functionality. **Stress testing** is a way to test reliability. **Load testing** is a way to test performance. There is little agreement on what the specific goals of load testing are. The terms load testing, performance testing, reliability testing, and volume testing, are often used interchangeably.

Software performance testing and load testing

- **Stability testing**

  Stability testing checks to see if the software can continuously function well in or above an acceptable period. This activity of non-functional software testing is often referred to as load (or endurance) testing.

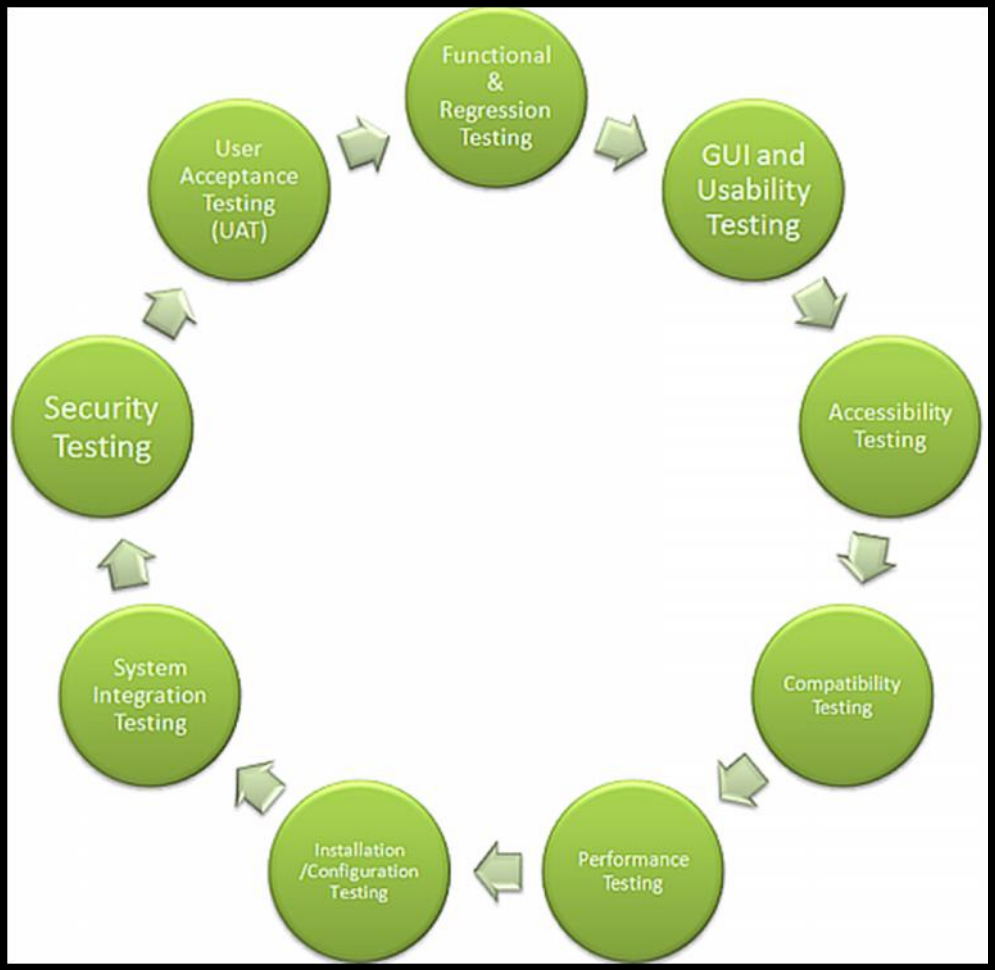
- **Usability testing**

  Usability testing is needed to check if the user interface is easy to use and understand. It is concerned mainly with the use of the application.

- **Security testing**

  Security testing is essential for software that processes confidential data to prevent system intrusion by hackers.

Software performance testing and load testing



METHODS OF TESTING

There are two basic methods of performing software testing:

1. Manual testing
2. Automated testing

MANUAL TESTING

As the name would imply, manual software testing is the process of an individual or individuals manually testing software. This can take the form of navigating user interfaces, submitting information, or even trying to hack the software or underlying database. As one might presume, manual software testing is labor-intensive and slow.

There are some things for which manual software testing is appropriate, including:

1. User interface or usability testing
2. Exploratory/ad hoc testing (where testers do not follow a 'script', but rather testers 'explore' the application and use their instincts to find bugs)
3. Testing areas of the application which experience a lot of change.
4. User acceptance testing (often, this can also be automated)

MANUAL TESTING

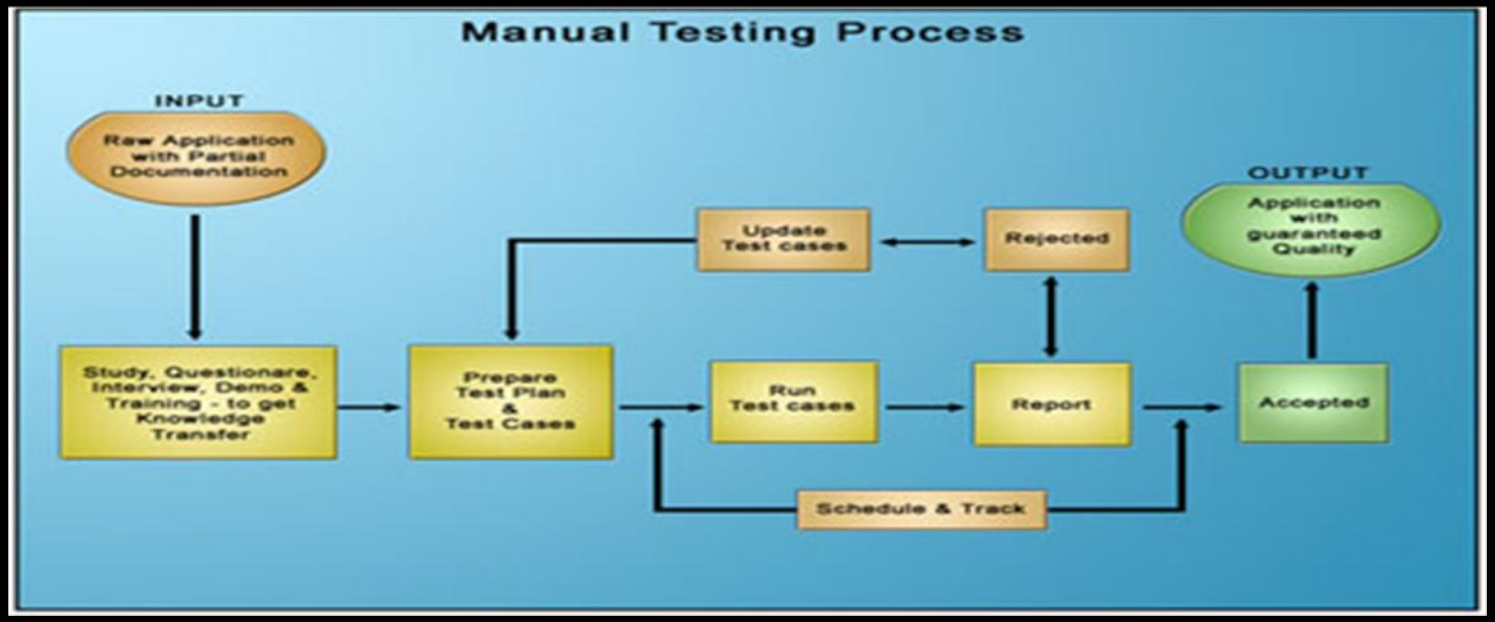
The time commitment involved with manual software testing is one of its most significant drawbacks.

The time needed to fully test the system will typically range from weeks to months. Variability of results depending on who is performing the tests can also be a problem. For these reasons, many companies look to automation as a means of accelerating the software testing process while minimizing the variability of results.

MANUAL TESTING

The following are the steps followed at Paradigm for Manual Testing;

1. Test Analysis
2. Test Planning
3. Test Case Design
4. Test Execution
5. Status Reporting with Pass/Fail
6. Regression test



AUTOMATED TESTING

Automated software testing is the process of creating test scripts that can then be run automatically, repetitively, and through many iterations. Done properly, automated software testing can help to minimize the variability of results, speed up the testing process, increase test coverage (the number of different things tested), and ultimately provide greater confidence in the quality of the software being tested.

There are, however, some things for which automated software testing is not appropriate. These include:

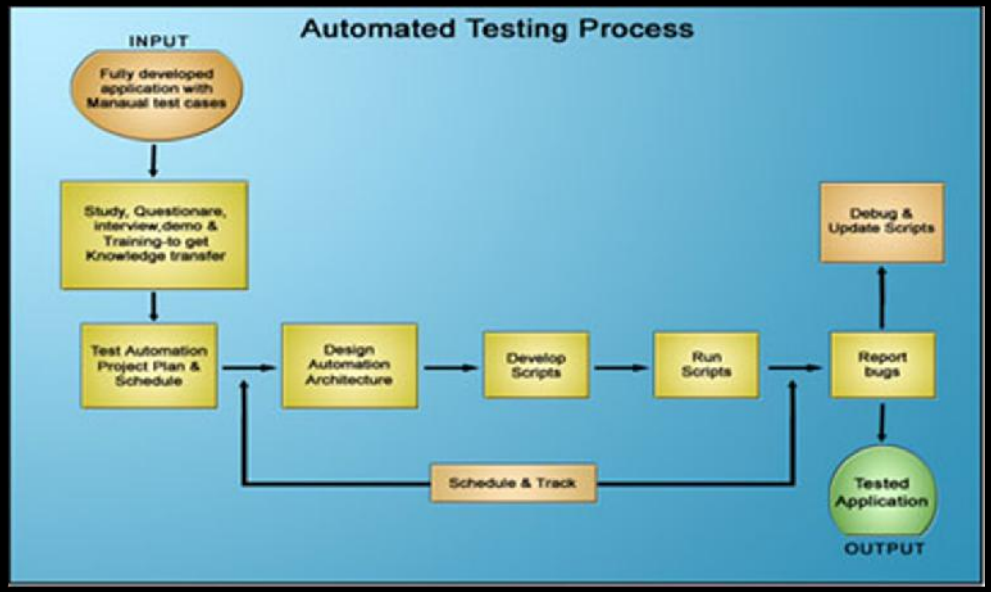
End user usability testing is not typically a good candidate for automated testing.

Tests for areas of the application which experience a lot of change are also not a good candidate for automation since this can lead to substantial maintenance of test automation scripts. Such areas of the application may be more effectively tested manually.

is important to note that test automation is software, and just like the software you are building for internal or external customers, it must be *well-architected*.

AUTOMATED TESTING

Automated testing is a valuable tool that provides us with additional ways of getting to the heart of the problem. Manually testing is a time consuming practice and is difficult to repeat. Using test automation tools to write programs that simulates thousands of executed commands in the same exact order. Each time that your software does not perform to your specifications, the program will record the exact command that caused the problem. Once you thought you fixed the problem, you could then run the very same set of commands to see if, in fact, you were successful.



Manual vs. Automated Testing

Manual tests are those that require human intervention to perform a test procedure. Automated tests can be designed to replicate any user/application activity in a high-speed and easily replicated environment. Not all tests should, or can be, automated. Certain processes that require human intervention, such as loading special paper into a printer, cannot be automated by a software tool.

Automation is a not a Replacement of Manual Testing :
Test automation is expensive and it is an addition, not a replacement, to manual testing. It can be made cost-effective in the longer term though, especially in regression testing.

What Tools Does :
Many test automation tools provide record and playback features that allow users to record interactively user actions and replay it back any number of times, comparing actual results to those expected.

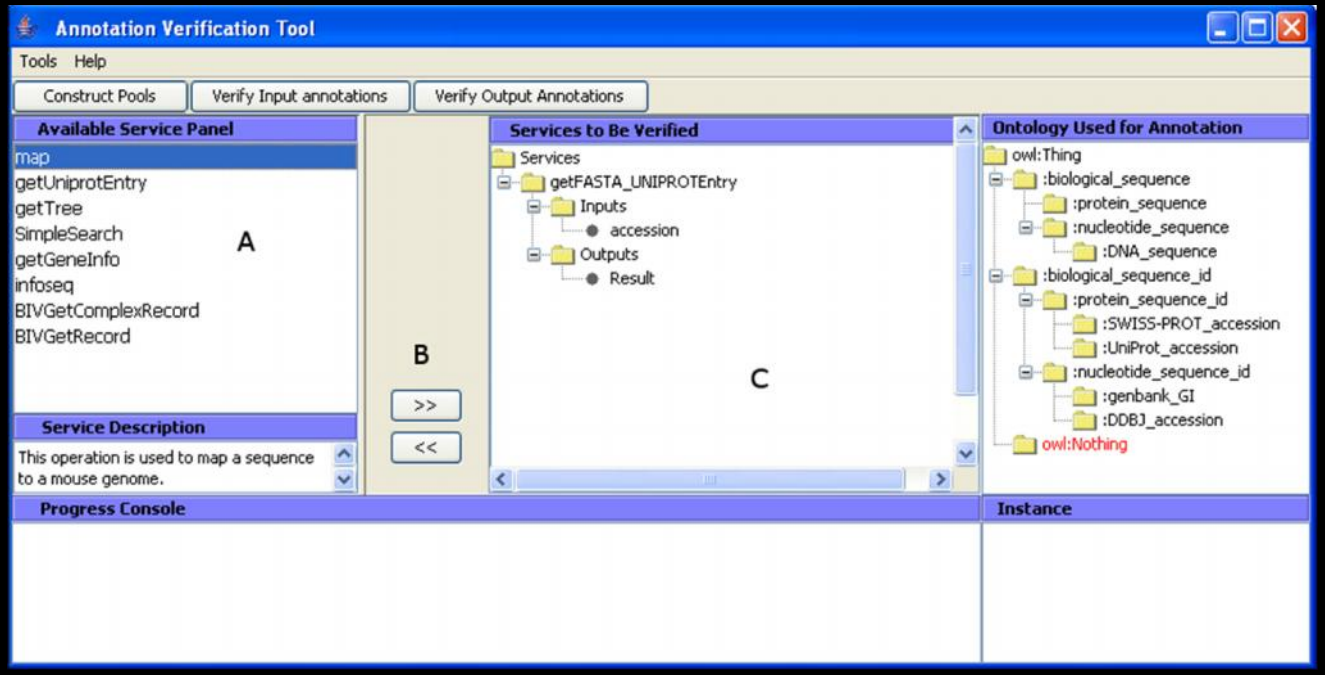
Can test automation improve test effectiveness?
Yes, Automating a test makes the test process:
1. Fast
2. Reliable
3. Repeatable
4. Programmable
5. Reusable

Black Box Testing Techniques

- ✓ Equivalence Class Partitioning Testing
- ✓ Boundary Value Testing
- ✓ Omission Testing
- ✓ Null Case Testing
- ✓ Volume Testing
- ✓ Load Testing
- ✓ Stress Testing
- ✓ Performance Testing
- ✓ Resource Testing
- ✓ Requirements/Specification Testing
- ✓ Button Press Testing
- ✓ State Transition Testing
- ✓ Installation Testing
- ✓ Security Testing
- ✓ Integration Testing
- ✓ Compatibility Testing
- ✓ Configuration Testing
- ✓ Documentation Testing
- ✓ Smoke Testing
- ✓ Usability Testing
- ✓ Exploratory Testing

Equivalence Partitioning

Equivalence Partitioning is a black-box testing method that divides the input domain of a program into classes of data from which test cases can be derived



DESIGNING THE TEST CASES

The next step is to design the test cases by drawing up a table showing the test case ID, a typical value drawn from the partition to be input, the partition it tests, and the expected output.

Test Case ID	Hotel Charge	Partition Tested	Expected Output
1	50	0< Hotel Charge<=70	OK
2	-25	Hotel Charge<=0	Error Message
3	89	Hotel Charge>70	Error Message

BOUNDARY VALUE TESTING

Boundary value analysis leads to a selection of test cases that exercise bounding values. This technique is developed because a great number of errors tend to occur at the boundary of input domain rather than at the center.

Test Case ID	Hotel Charge	Boundary Tested	Expected Output
1	-1	0	Error Message
2	0		Error Message
3	1		OK
4	69	70	OK
5	70		OK
6	71		Error Message

Omission Testing

- **Omission Testing (also called Missing Case Testing):**
- Exposes defects caused inputting cases (scenarios) the developer forgot to handle or did not anticipate.

NULL Testing

- **Null Testing: (a specific case of Omission Testing, but triggers defects extremely often)**
- Exposes defects triggered by no data or missing data.
- Often triggers defects because developers create programs to act upon data, they don't think of the case where the project may not contain specific data types.

Volume Testing

- Volume testing is done against the efficiency of the application. Huge amount of data is processed through the application (which is being tested) in order to check the extreme limitations of the system.
- The purpose of Volume Testing is to find weaknesses in the system with respect to its handling of large amounts of data during short time periods
- Such systems can be transactions processing systems capturing real time sales or could be database updates and or data retrieval.
- Volume testing will seek to verify the physical and logical limits to a system's capacity and ascertain whether such limits are acceptable to meet the projected capacity of the organization’s business processing.

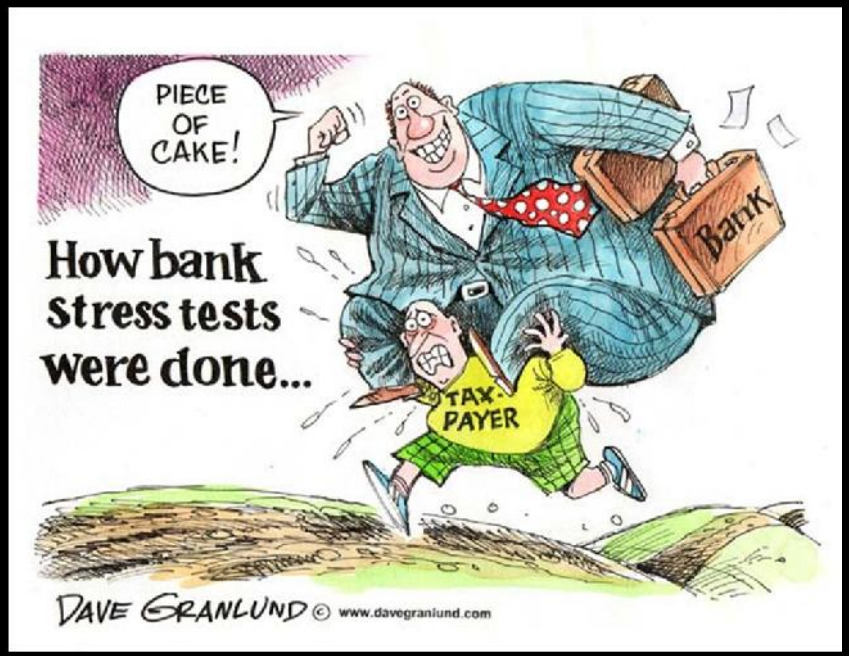
Load Testing

- Exposes defects triggered by peak bursts of activity.

Load Testing

Stress Testing

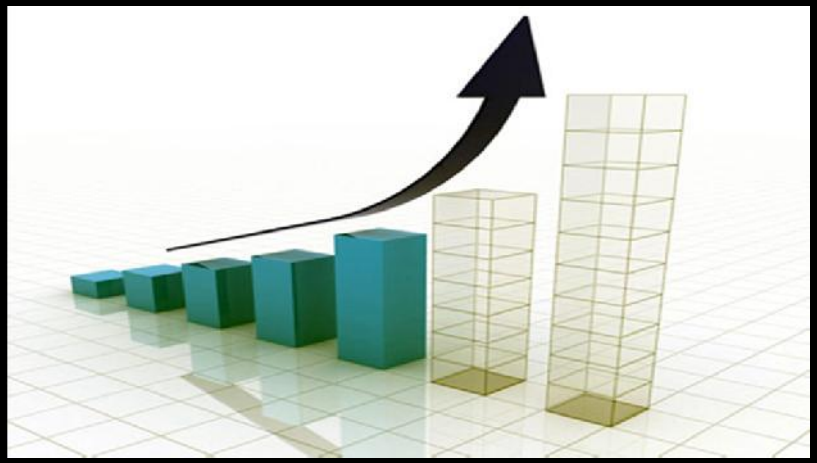
- In stress testing you continually put excessive load on the system until the system crashes
- A test environment is established with many testing stations. At each station, a script is exercising the system.



Performance Testing

- Exposes defects related to tasks taking too long.
- Definition of “taking too long” should be:
- Slower than specifications listed in the Requirements or Specification Document
- Slower than the previous release performing the same task on same data and machine.
- Slower than a client would reasonably expect. Such as if it takes 3 minutes to display a normal image clients will not use the product.
- Performance testing is designed to test the run time performance of a software

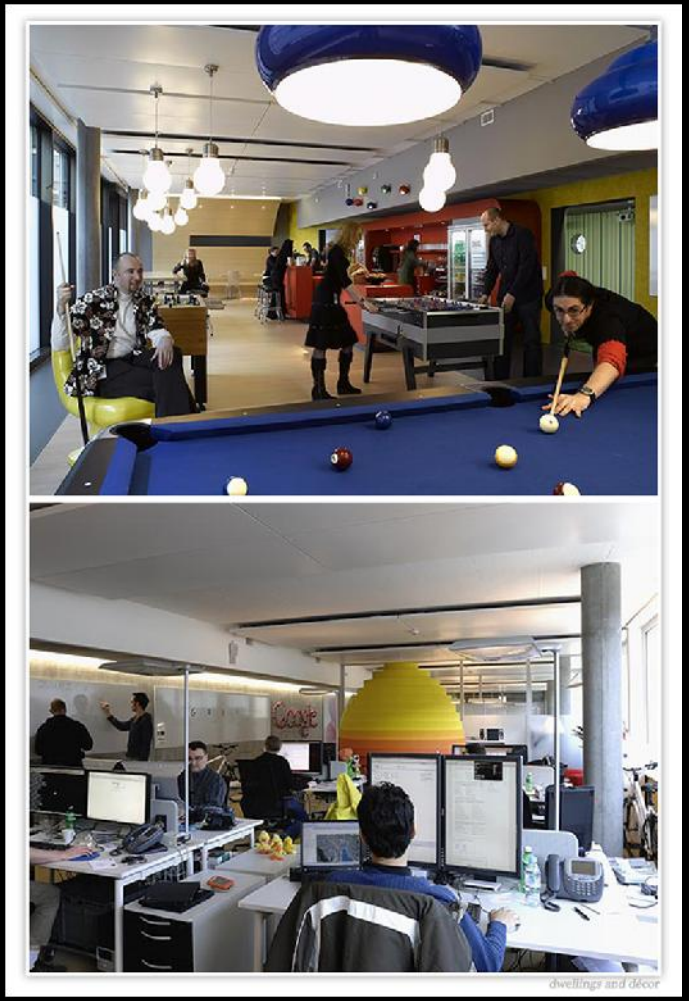
Performance Testing



Resource Testing

- In resource testing you have to check whether an AUT(Application under test) utilizes more resources (e.g memory) than it should be utilized

Resource Testing



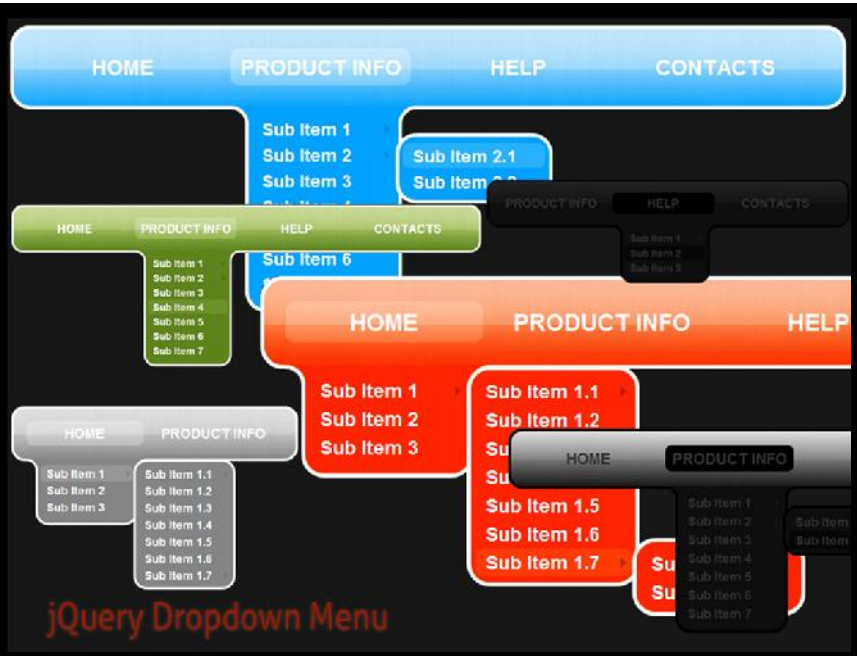
Resource Testing

Requirements Testing

- **Requirements or Specification Testing**
- Exposes defects in the program design/implementation by comparing the program to every word in the Requirement Document or the Function Specification Document. Important these documents kept up-to-date.

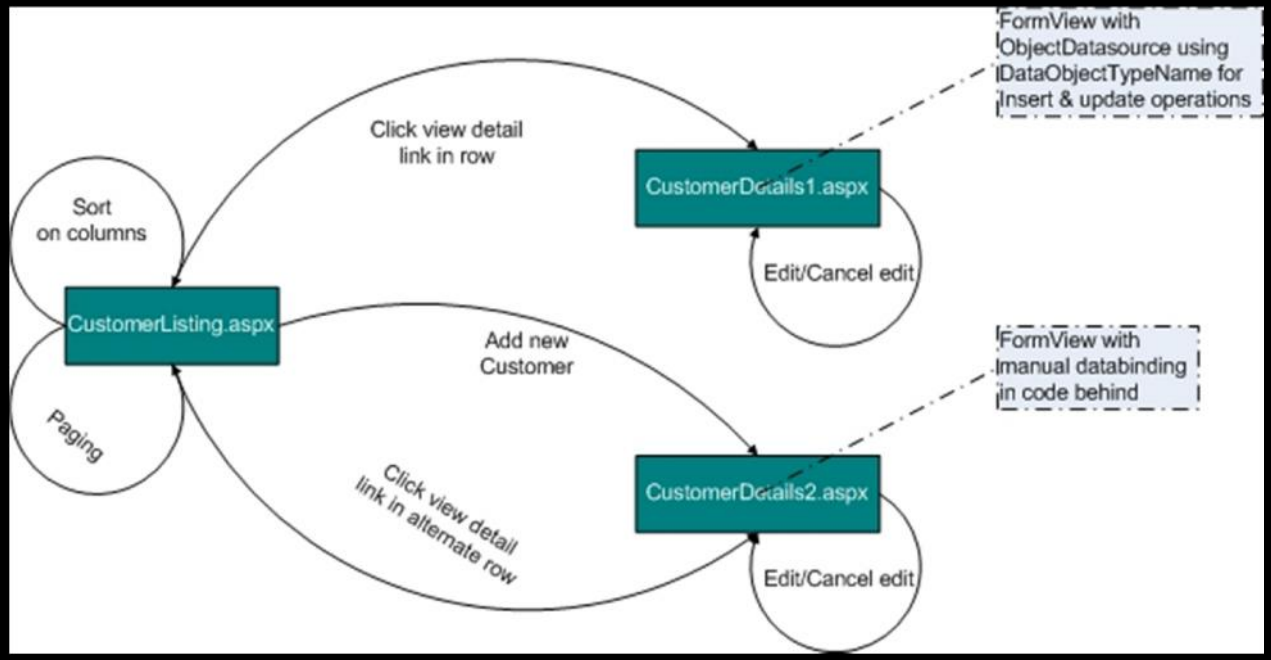
Button Press

- **Button Press Testing: (Landmark testing term, not industry standard)**
- Exposes functionality defects by methodically pressing every widget (pull down menu, pop ups, drop down lists, buttons, icons, etc.) in the program.



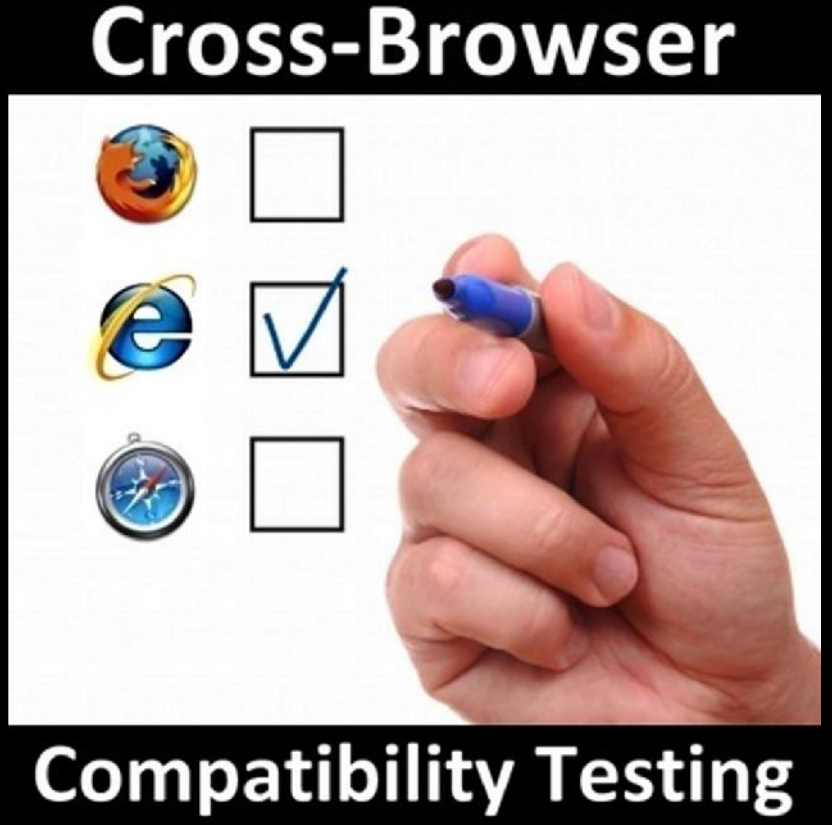
State Transition Testing

- State transition testing is used where some aspect of the system can be described in what is called a “finite state machine”.
- This simply means that the system can be in a (finite) number of different states, and the transitions from one state to another are determined by the rules of the “machine”.



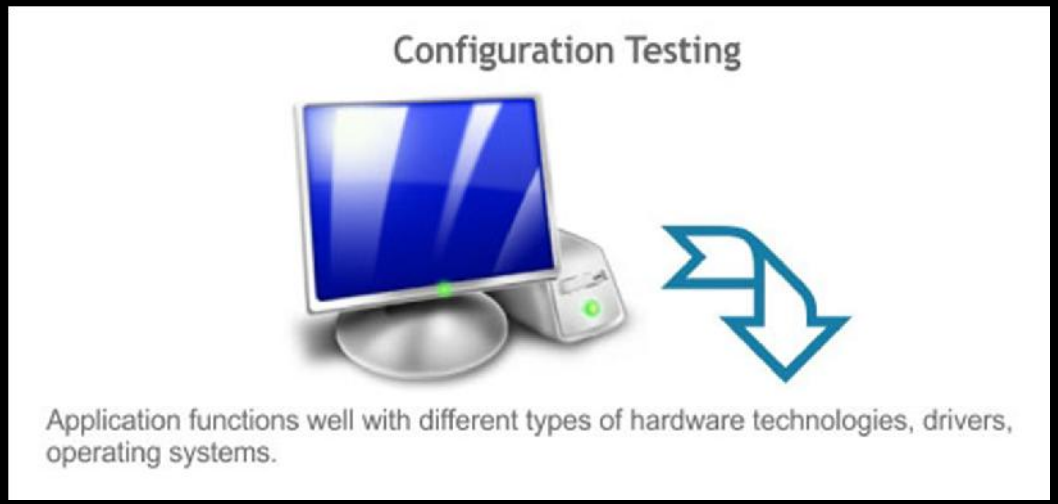
Compatibility Testing

- Exposes defects related to using files from output one version of the software in another version of the software.



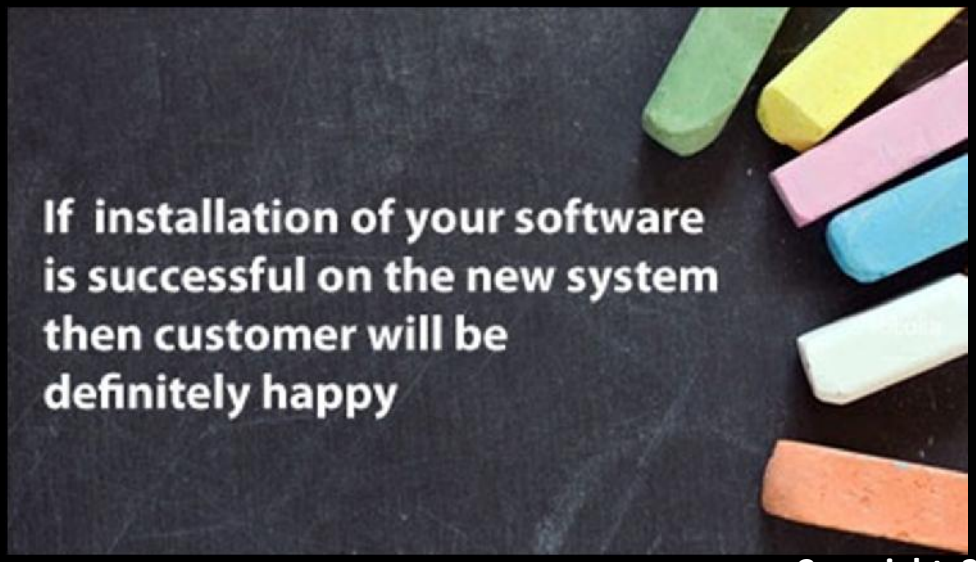
Configuration testing

- Configuration testing is also called "Hardware compatibility testing“ or portable testing , during this testing tester will test whether the software build is supporting different hardware technologies or not Ex : printer, Scanners e.t.c
- It is impossible to test all environments so must concentrate on those minimum and target environments.
- Exposes defects triggered by different computer environments.



Installation Testing

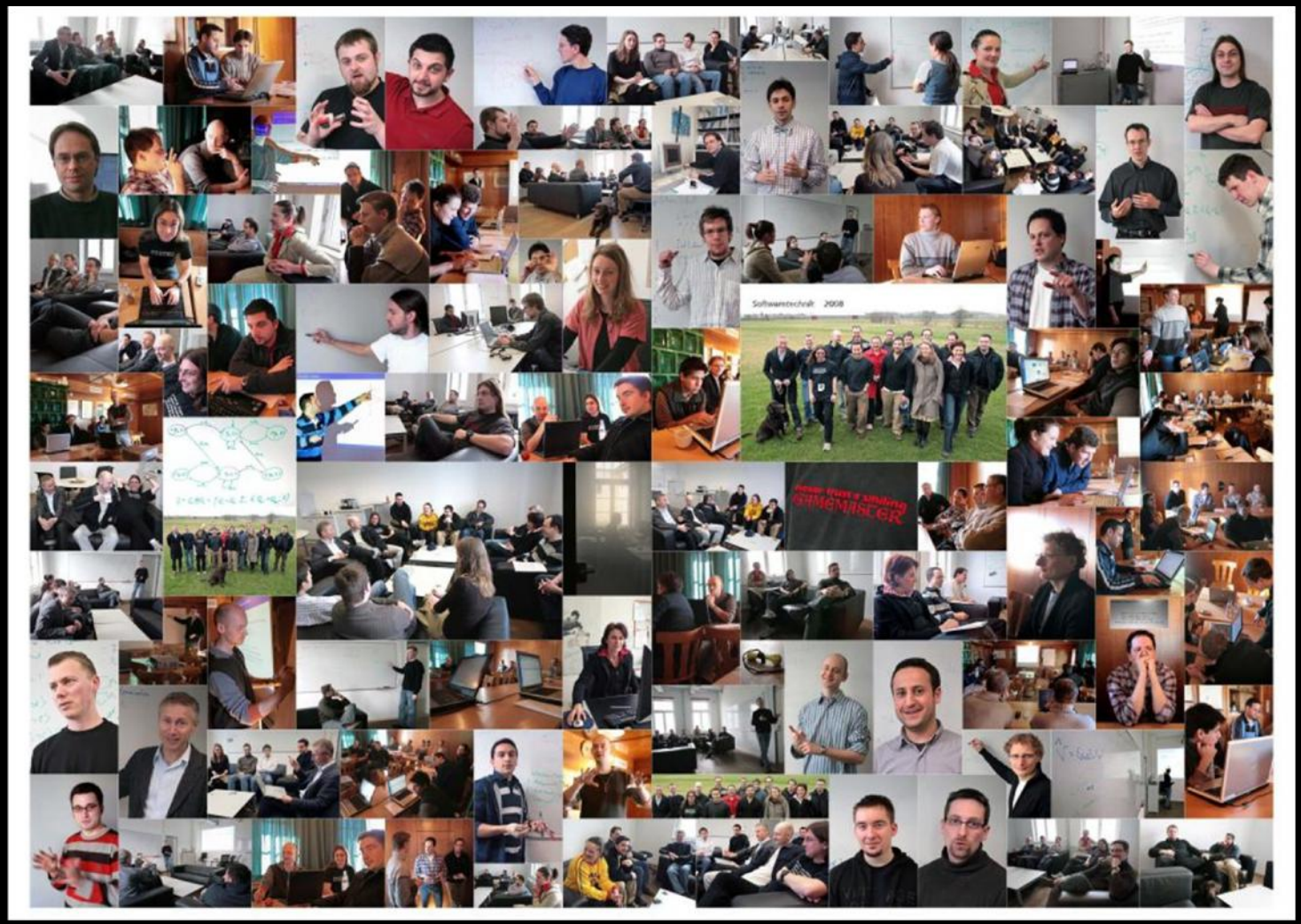
- Installation Testing refers to testing of full or partial steps of installation process.
- Installation testing is a kind of quality assurance work in the software industry that focuses on what customers will need to do to install and set up the new software successfully.



Security Testing

- Security Testing refers to testing how well the system protects against unauthorized internal or external access.
- Security Testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration .

Security Testing



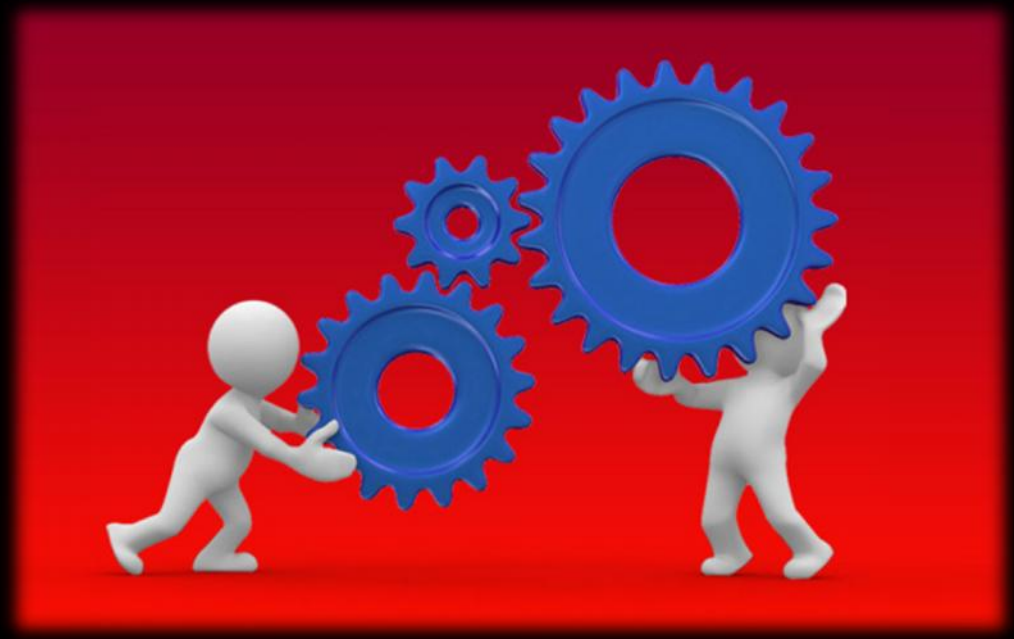
Integration Testing

Integration Testing is a systematic technique for constructing a program structure while at the same time conducting tests to uncovers errors associated with interfacing.

- There are two main approaches of integration testing.

✓ **Big Bang Integration**

✓ **Incremental Integration**



Big Bang Integration

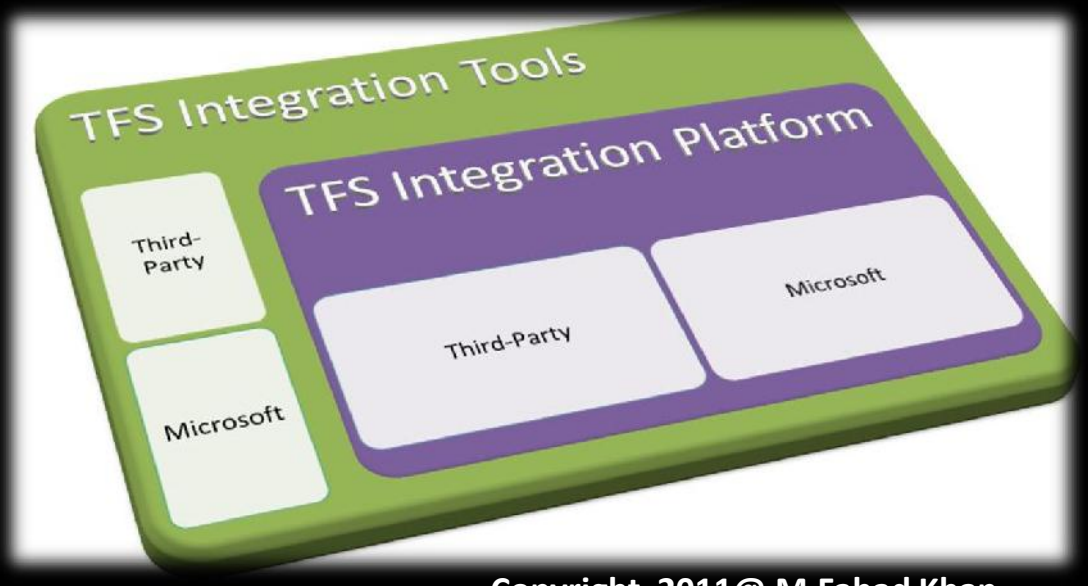
- There is often a tendency to attempt non incremental integration; that is, to construct the program using a big bang approach.
- All components are combined in advance. The entire program is tested as a whole. A set of errors is encountered.
- Correction is difficult because isolation of causes is complicated by the vast expense of the entire program. Once these errors are corrected new ones appear and the process continues in a seemingly endless loop.

Incremental Integration

In incremental integration the program is constructed and tested in small increments, where errors are easier to isolate and correct; interfaces are more likely to be tested completely. *Team Foundation Server Integration Tools* (TFS) is working on it.

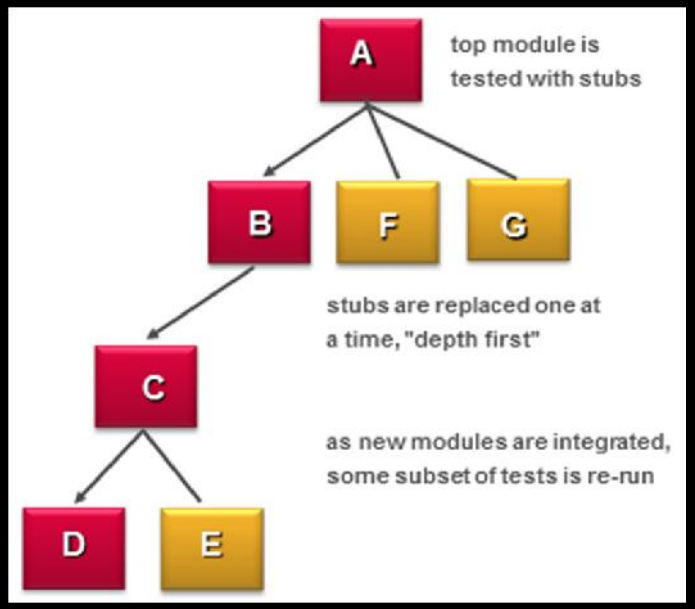
There are two types of integration testing::

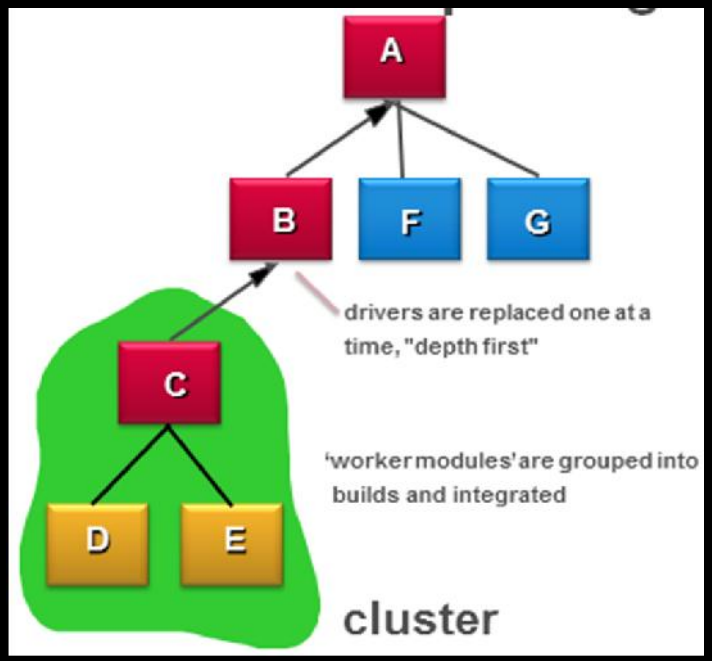
- ✓ Top Down Integration
- ✓ Bottom Up Integration



Top Down Integration

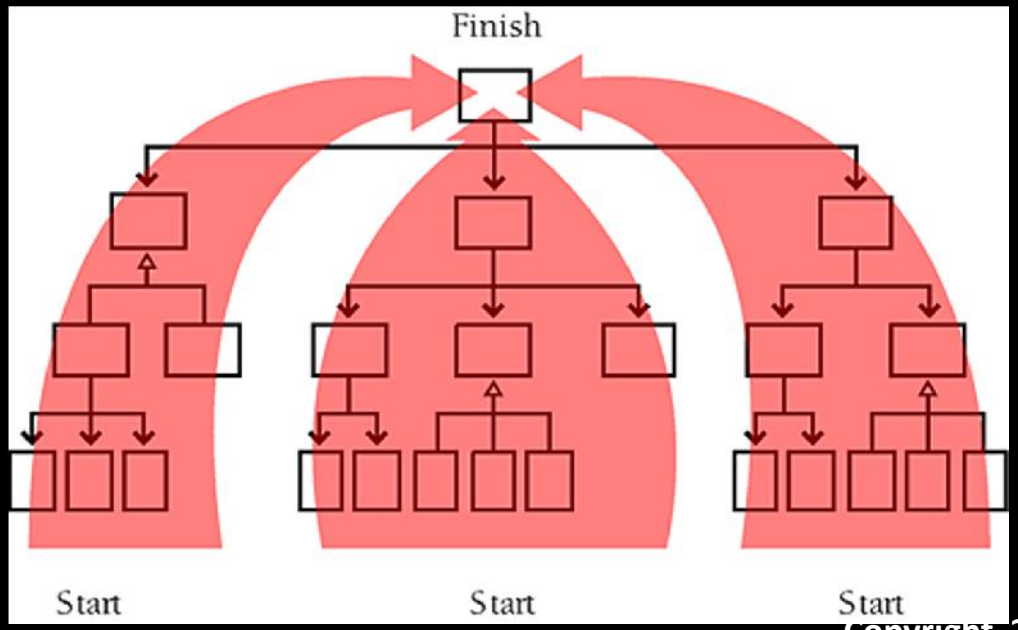
Top down integration is an incremental approach to construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control module (main program). Modules subordinate to the main module are incorporated into the structure in either a depth-first or breadth-first manner.





Bottom Up Integration

Bottom up integration as the name implies begins constructing and testing with atomic modules. Because components are integrated in bottom up fashion, processing required for components subordinate to a given level is always available and the need for stub is eliminated.



Documentation Testing

- Exposes defects in the content and access of on-line user manuals (Help files) and content of training manuals.
- The Testing Group tests that all Help files appear on the screen when selected.

Phases of Documentation Testing

- 1st Phase is Review and Inspection, examines the documents for editorial clarity.
- 2nd phase is Live Test, which uses the documentation in conjunction with the use of the actual program.

Smoke Testing

- This type of testing is also called sanity testing. A test of new or repaired equipment by turning it on. If it smokes... guess what... it doesn't work! . The term was originally coined in the manufacture of containers and pipes, where smoke was introduced to determine if there were any leaks.
- Smoke testing also known as build verification testing: A relatively small suite of tests is used to qualify a new build. Normally, the tester is asking whether any components are so obviously or badly broken or some critical fixes that are the primary intent of the new build didn't work. The typical result of a failed smoke test is rejection of the build not just a new set of bug reports.
- Smoke tests are designed to confirm that changes in the code function as expected and do not destabilize an entire build.

Usability Testing

- Exposes operations that are difficult, awkward, or inconvenient for users.
- Testing for “User-friendliness”
- Clearly this is subjective and will depend on the targeted end user
- User interviews, surveys, video recording of user sessions, and other techniques can be used

Usability Testing

Regression Testing

- Exposes defects in code that should have not changed.
- Re-executes some or all existing test cases to exercise code that was tested in a previous release or previous test cycle.
- Performed when previously tested code has been re-linked such as when:
- Ported to a new operating system
- A fix has been made to a specific part of the code.
- A fix has been made to another part of the code, but this module had to be re-linked because the fix was in an underlying library this module also uses.
- Chances are defects may not be fixed correctly or the code change may introduce new defects.

Exploratory Testing

- **Exploratory testing is a method of manual testing.**
- Exploratory testing seeks to find out how the software actually works, and to ask questions about how it will handle difficult and easy cases.
- The testing is dependent on the tester's skill of inventing test cases and finding defects. The more the tester knows about the product and different test methods, the better the testing will be.

Recovery Testing

Recovery testing is a system test that forces the system to fall in a variety of ways and verifies that recovery is properly performed. If recovery is automatic then re-initialization, check pointing mechanisms and data recovery are evaluated for correctness. If recovery requires human intervention, the MTTR is evaluated to determine whether it is within acceptable limits.

If you have any query please feel free to ask

Phone: +92-51-9047-592

Fax: +92-51-9047-420

Email: fahad.khan@uettaxila.edu.pk

University Of Engineering & Technology Taxila Pakistan