Software Quality Assurance & Test

LECTURE # 13

SOFTWARE TESTING (Dynamic)

chenbo@etao.net

Topics to Cover

- Software Testing
- Principles of Testing
- Software Testing Lifecycle
- Software Testing Activities
- Role of Tester
- Skills of Tester
- Testing Methods
- SQA Team
- Testing Stages
- Test Cases
- Testing Types

Software Testing

- "Testing is the process of executing a program or system with the intent of finding errors." by Myers 1979
- Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item (IEEE, 1986; IEEE, 1990).

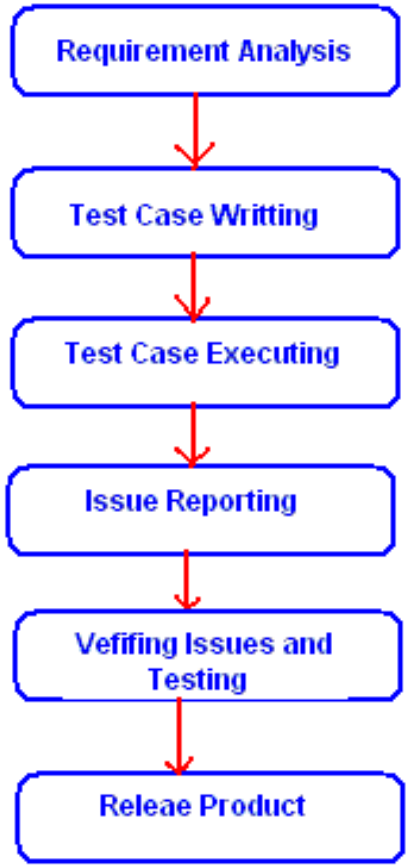
Principles of Testing

- Glenford Myers in his book "The Art of Software Testing" suggested the following testing principles
  - ✓ A necessary part of a test case is a definition of the expected output or result.
  - ✓ A programmer should avoid attempting to test his or her own program.
  - ✓ Thoroughly inspect the results of each test.
  - ✓ Test cases must be written for input conditions that are invalid and unexpected, as well as for those that are valid and expected.

Static Testing Approach

- Examining a program to see if it does not do what it is supposed to do is only half the battle; the other half is seeing whether the program does what it is not supposed to do.
- Avoid throwaway test cases unless the program is truly a throwaway program.
- Do not plan a testing effort under the assumption that no errors will be found.
- The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section.
- Testing is an extremely creative and intellectually challenging task.
- Exhaustive testing is not possible but we can assure that all conditions have been exercised

Testing Life Cycle



Testing Perspective

- *Developer*
  - **Understands the system but will test "gently" and is driven by "delivery"**
- *Independent Tester*
  - **Must learn about the system but will attempt to break it and is driven by quality**

Generic Testing Process

1. Test Planning:
   - 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
2. Test Design and Specification:
   - Conduct software design based well-defined test generation methods. Specify test cases to achieve a targeted test coverage.
3. Test Set up:
   - Testing Lab Space and tools (Environment Set-up) Test Suite Set-up
4. Test Operation and Execution:
   - Run test cases manually or automatically
5. Test Result Analysis and Reporting:
   - Report software testing results and conduct test result analysis

Generic Testing Process

6. Problem Reporting:
   - Report program errors using a systematic solution.
7. Test Management and Measurement:
   - Manage software testing activities, control testing schedule, measure testing complexity and cost
8. Test Automation:
   - Define software test tools
   - Adopt and use software test tools
   - Write software test scripts and facility
9. Test Configuration Management
   - Manage and maintain different versions of software test suites, test environment and tools, and documents for various product versions.

Role of Software Tester in SDLC

- A Software tester is required from early stages of SDLC

  - In Requirements Phase -- Tester does requirement Analysis
  - In Design Phase -- Tester does use case analysis and start drafting test plan
  - In Development phase -- Tester starts developing test cases and test script and finalize the test plan
  - In Testing Phase -- Execute various test cases and maintain logs, test summary reports and bug reports etc
  - In Deployment Phase -- prepares training documentation, lessons learned etc

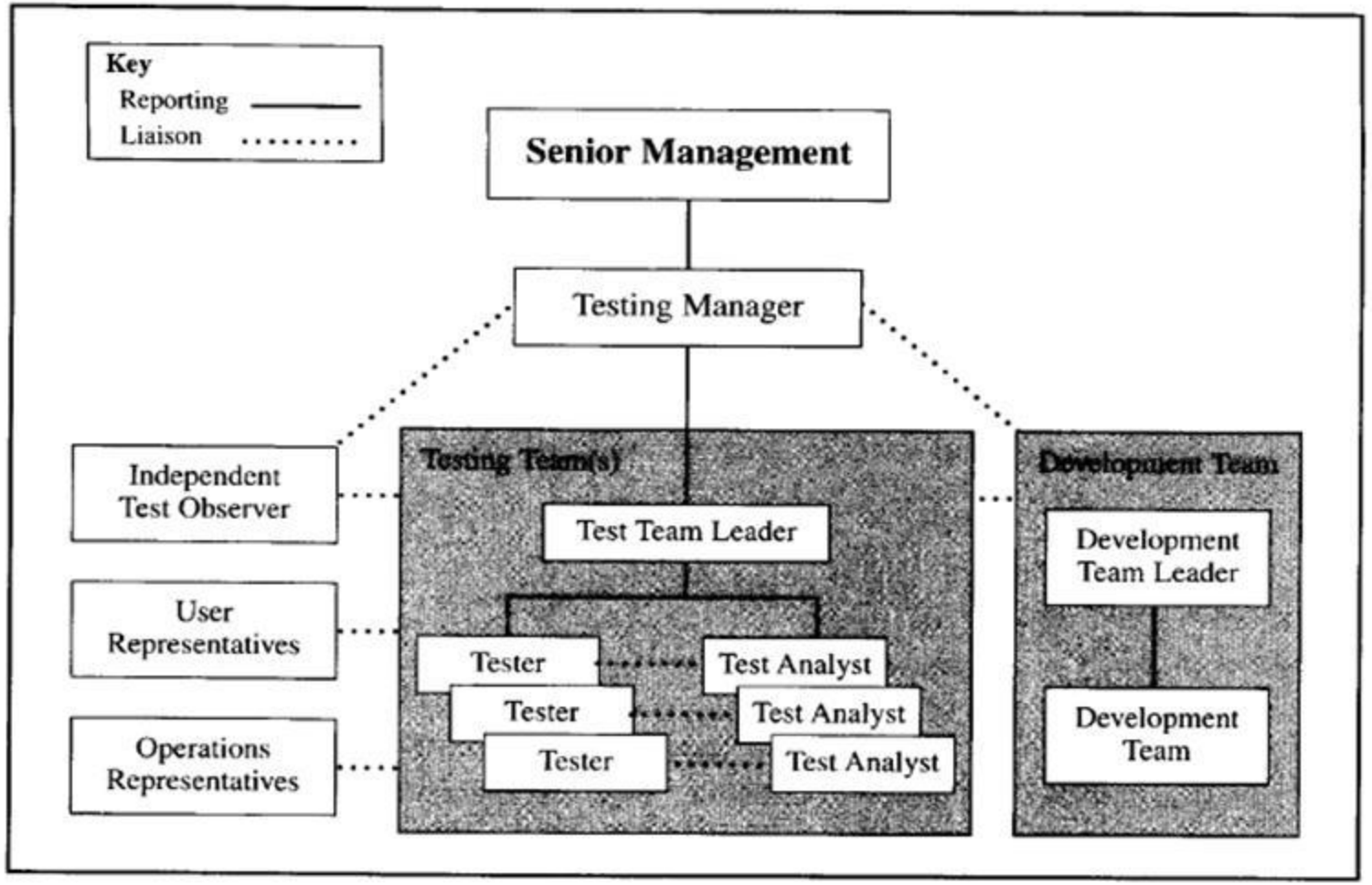
Skills for Software Tester

- Software Engineering Background
- Knowledge of Testing Tools
- Test Planning, test case development and test scripts execution skills
- Good Analytical Skills
- Good Communication and conflict management skills to work with developers and other project team members

SQA Team

- The Software Quality Assurance group can include the following Professionals
  - **Testing Manager**
  - **Test Team Lead**
  - **Test Analyst**
  - **Tester**
  - **Independent Test Observer**

SQA Team

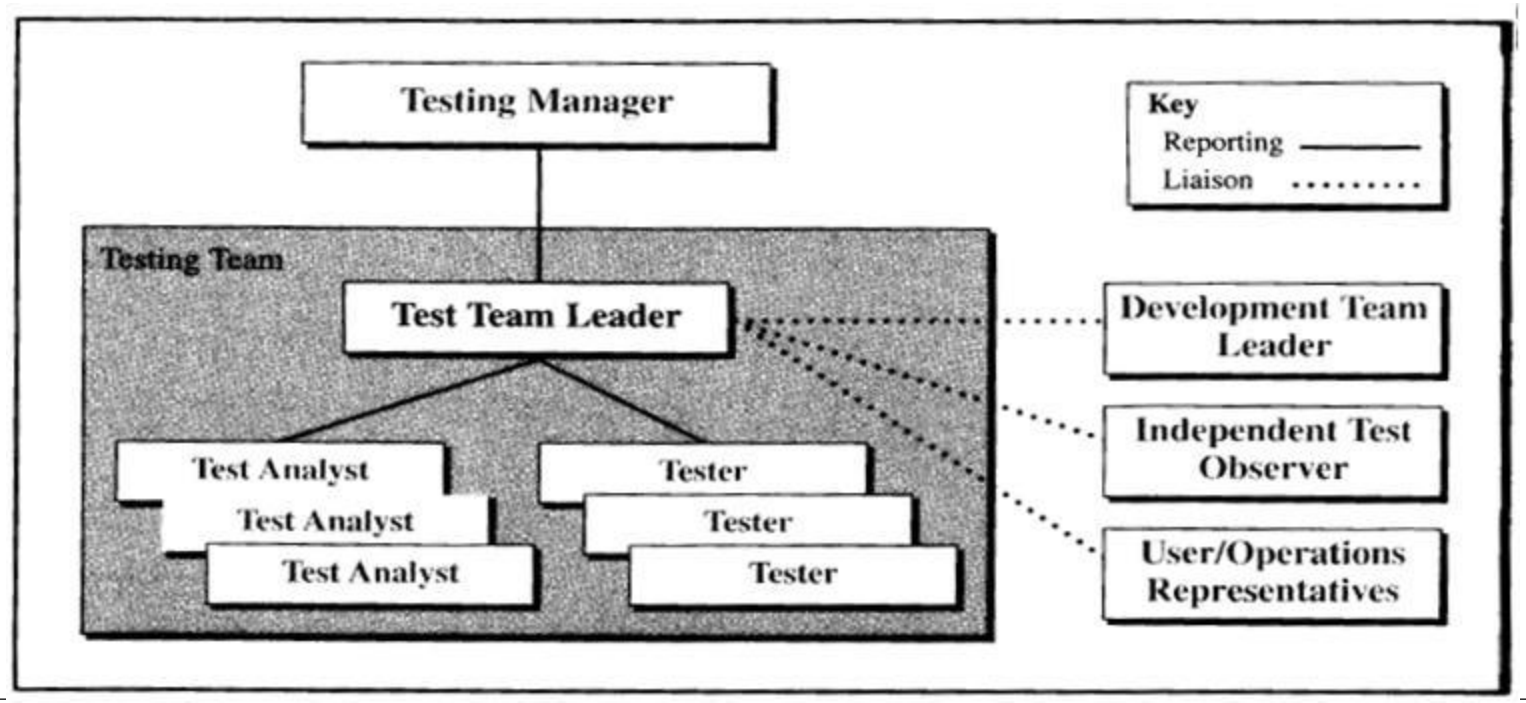

Key
Reporting
Liaison
Senior Management
Testing Manager
Independent Test Observer
User Representatives
Operations Representatives
Testing Team(s)
Test Team Leader
Tester
Test Analyst
Tester
Test Analyst
Tester
Test Analyst
Development Team
Development Team Leader
Development Team

SQA Team

- **Testing Manager**
  - The testing manager has the authority to monitor/administer the organizational aspects of the testing process on a day-to-day basis and is responsible for ensuring that the individual testing projects produce the required products to the required standard of quality & within the specified constraints of time, resources and costs.
  - Test manager is also responsible for liaison with the development teams to ensure that they follow the unit and integration testing approach documented within the process.
  - Test manager is also responsible for liaison with the Independent Test Observer to receive reports on testing projects.
  - Test manager reports to a senior manager.

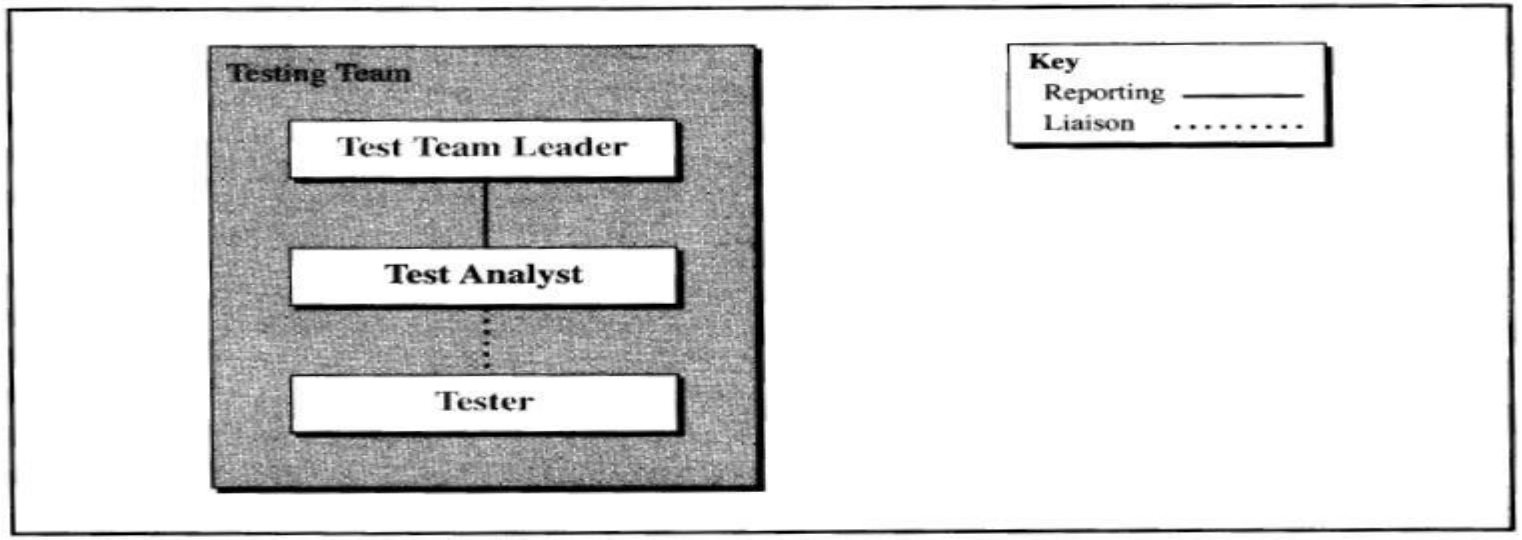
SQA Team

- **Testing Team Lead**
  - The testing team lead is given the authority to run a testing project.
  - His or her responsibility includes tasking one or more Test Analysts and Testers, monitoring their progress against agreed –upon plans, setting up and maintaining the testing project and ensuring the generation of the testing project artifacts.



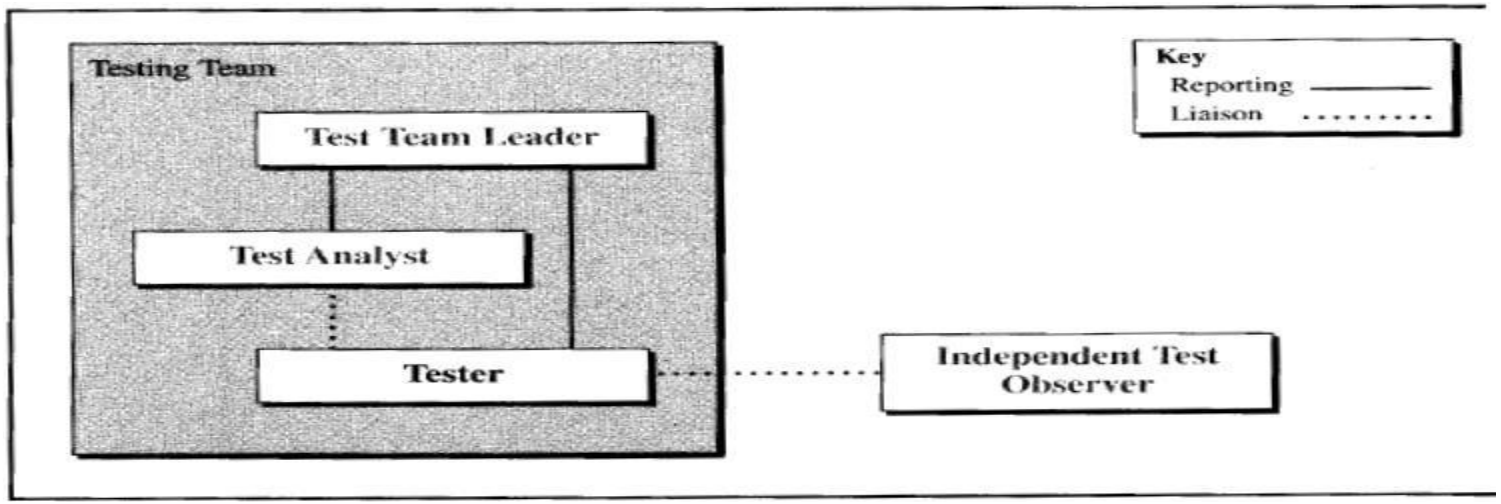
SQA Team

- **Testing Analyst**
- The testing analyst is responsible for the design and implementation of test cases which will be used to accomplish the testing of AUT
- The Test Analyst may also be called upon to assist the Team Lead in the generation of test specification document.



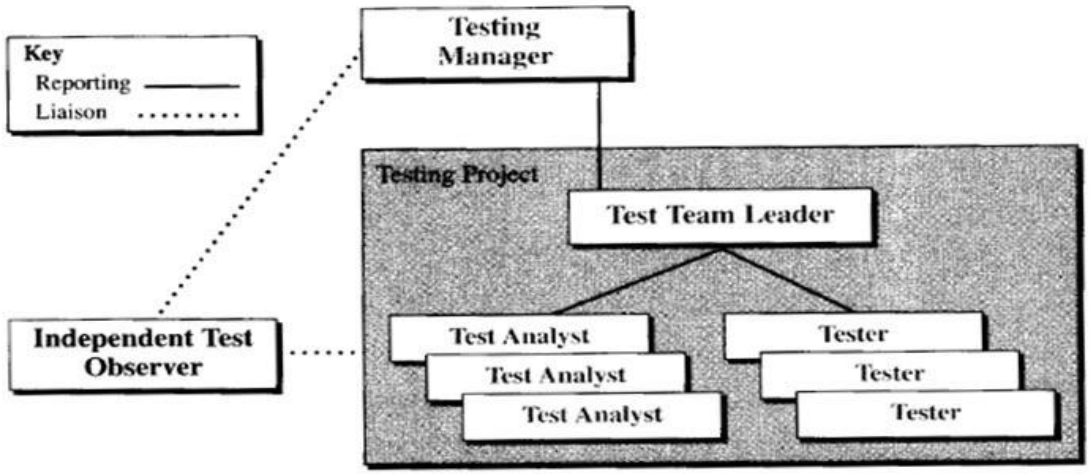
SQA Team

- **Tester**
  - The tester is responsible for the execution of test cases created by test analyst or sometimes by tester himself. And for the interpretation and documentation of the results of the test cases
  - Prior to the execution of the test cases, the Tester will set up and initialize the test environment, including the test data, plus any additional software required to support the test.



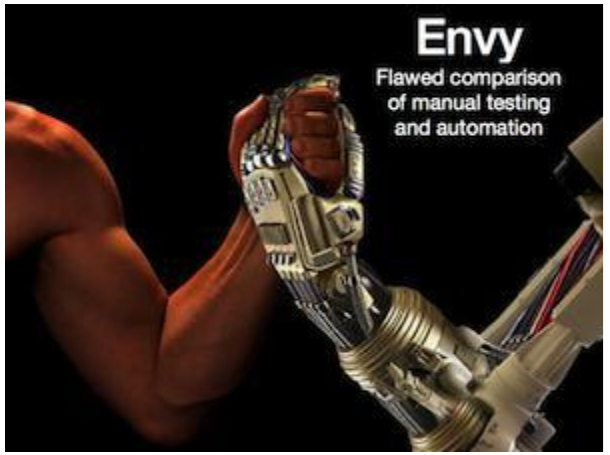
Meeting less Inspection

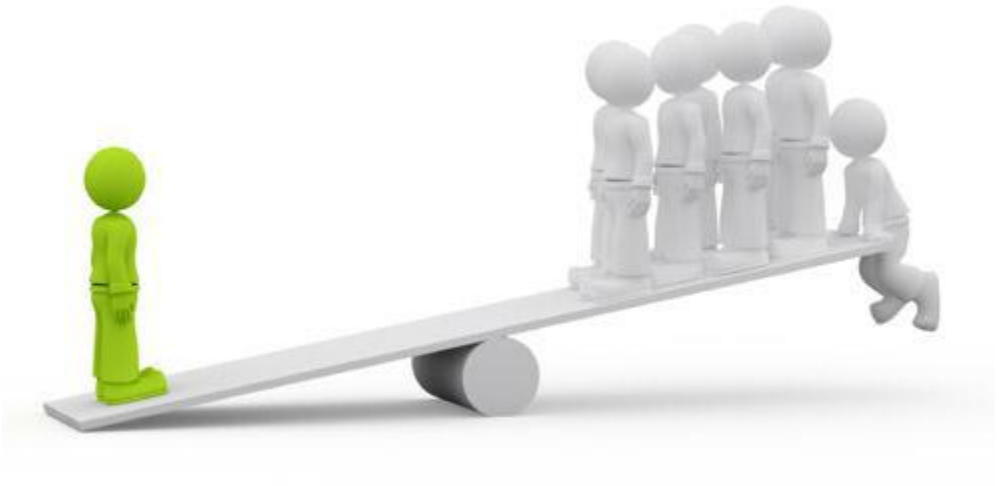
- **Independent Test Observer**
  - Independent test observers can be hired or invited to provide the independent observation of the testing project
  - Independent test observer is responsible for:
    - **Attending the testing of AUT**
    - **Formally witnessing that the tester correctly implements all test cases**
    - **Signing the appropriate section of Test Result Record Form**
    - **Liaising with the Testing manager to report any problems observed during the testing process**



Testing Methods

- Manual Testing
- Automated Testing

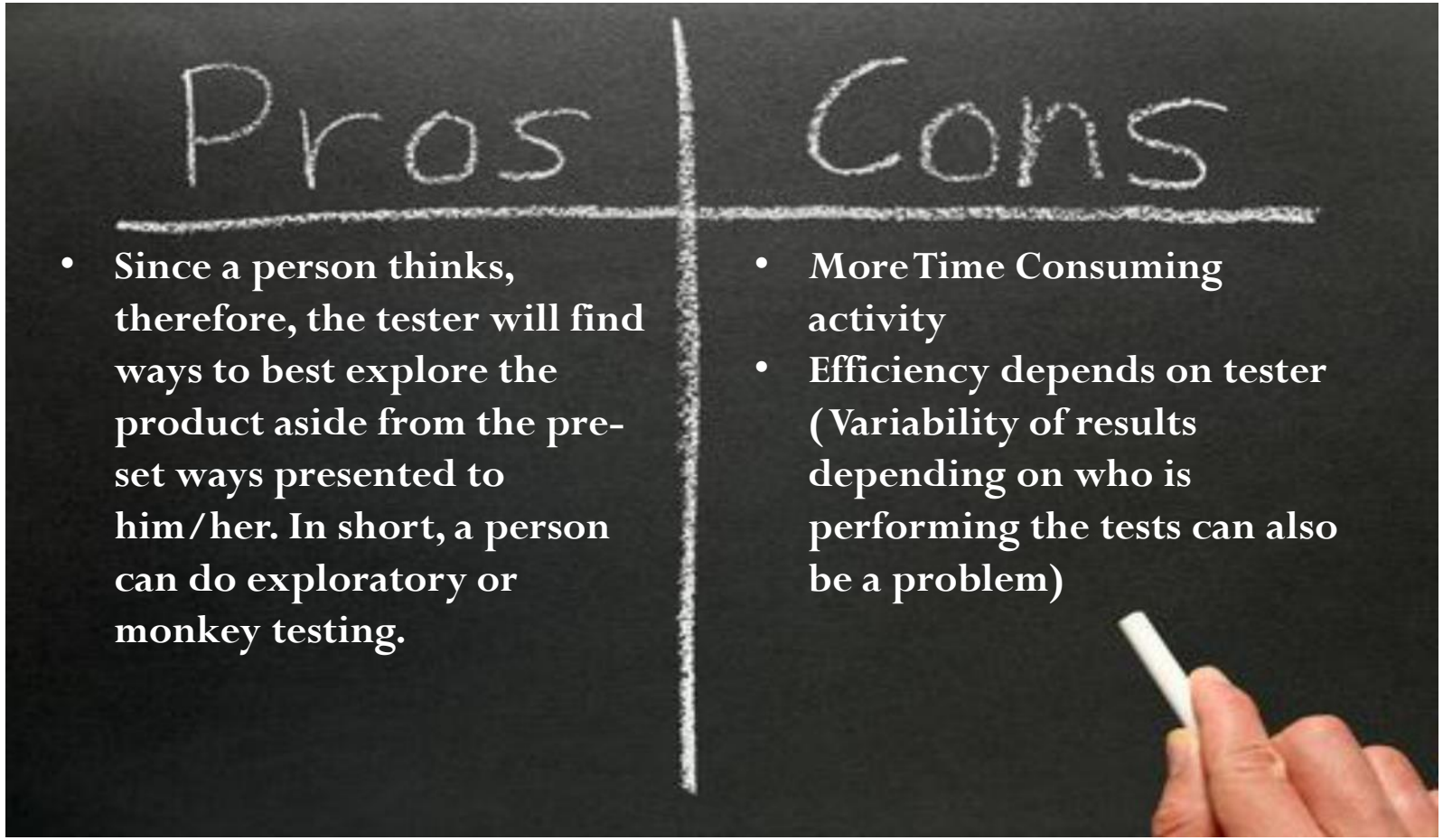




Manual Testing

- This type includes the testing of the Software manually i.e. without using any automated tool or any script.
- In this type the tester takes over the role of an end user and test the Software to identify any un-expected behavior or bug.
- There are different stages for manual testing like unit testing, Integration testing, System testing and User Acceptance testing.
- Testers use test plan, test cases or test scenarios to test the Software to ensure the completeness of testing.
- Manual testing also includes exploratory testing as testers explore the software to identify errors in it.

Manual Testing



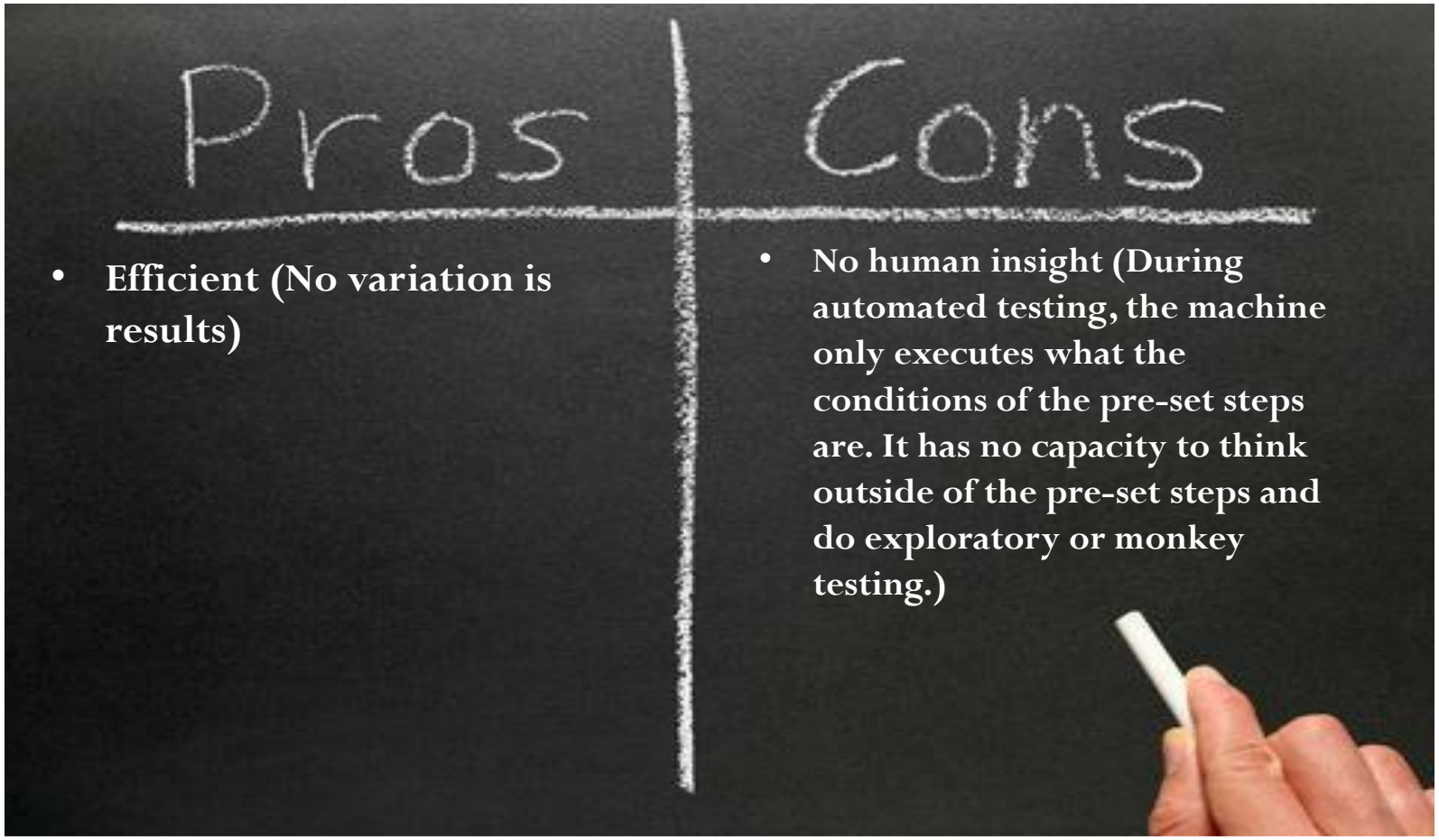
Manual Testing

- Condition: Manual tests can be used in situations where the steps cannot be automated, for example to determine a component's behavior when network connectivity is lost; this test could be performed by manually unplugging the network cable.
- In real time 60 to 70 % testing done manually. Tester will create test cases, Tester will execute test cases, tester will write bug report manually.
- Except **performance testing** and **Stress Testing** every thing we can do manually.
- Examples of manual Testing can be:
  - Application Usability Check
  - Documentation check

Automated Testing

- Automation testing, also known as *Test Automation*, is when the tester writes scripts and uses another software to test the software.
- This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly and repeatedly.
- Automation is a not a Replacement of Manual Testing
- Done properly, automated software testing can help
  - to minimize the variability of results,
  - speed up the testing process,
  - increase test coverage, and
  - ultimately provide greater confidence in the quality of the software being tested.

Over the Shoulder Reviews

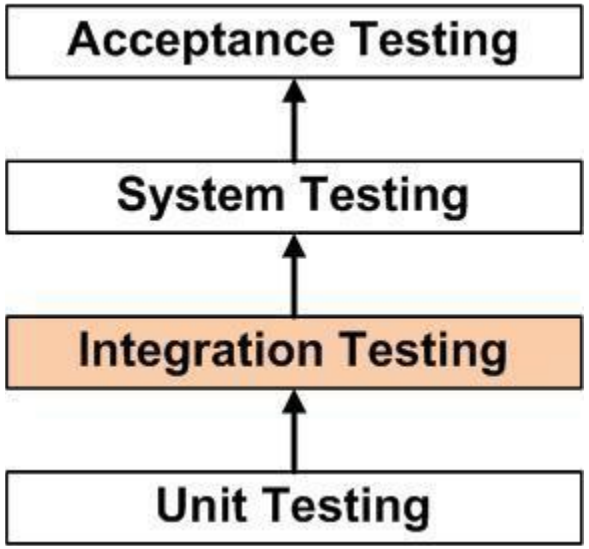


Manual vs Automated Testing

MANUAL TESTING	AUTOMATED TESTING
Time consuming and tedious	Fast
Huge investment in human resources	Less investment in human resources
Less reliable	More reliable
Non-programmable	Programmable
Manual Test is not reusable	Automated Tests are reusable
Manual Test provide limited visibility	Automated Test provide global visibility
High risk on missing out something	No risk of missing out something

Stages of Testing

- **Unit testing/Component Testing/Module testing[2,3,6]**
- **Integration Testing**
- **System testing**
- **Acceptance testing**



Test Case

- A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system meets specifications.
- A test case has components that describes an input, action or event and an expected response, to determine if a feature of an application is working correctly.
- The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle.
- Test Suite: A collection of test cases.
- A test case may include many subsets.

Test Case

- Test Case ID: It is unique number given to test case in order to be identified.
- Test description: The description of test case you are going to test.
- Revision history: Each test case has to have its revision history in order to know when and by whom it is created or modified.
- Function to be tested: The name of function to be tested.
- Test Setup: Anything you need to set up outside of your application for example printers, network and so on.

Test Case

- Test Execution: It is detailed description of every step of execution.
- Expected Results: The description of what you expect the function to do.
- Actual Results: pass / failed
  - ✓ If pass - What actually happen when you run the test.
  - ✓ If failed - put in description of what you've observed.

Test Case

- Test case for login creation? User name must be of type character and password must be minimum of length 6.

Case #	Test case	Test Execution	Expected Results	Pass/ Fail	Remarks
1	Login creation	Enter abcdef in user name field Enter the password, length of 6 or more Enter the same password in confirm password field Press enter	Login page created		
2		Enter abcdef in user name field Enter the password, length of 6 or more Enter some other password in confirm password field Press enter	Error message if check is there for confirm password		Validator check should be visible suggesting that passwords not matched

Traceability Matrix

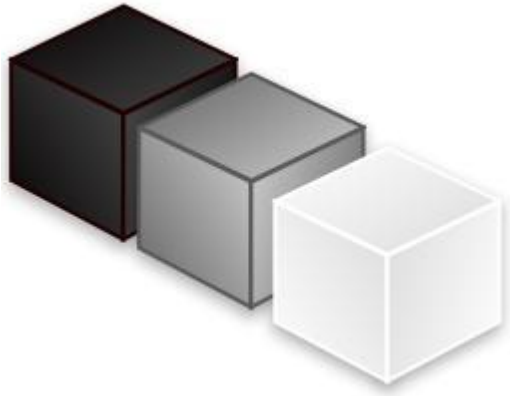
- Traceability Matrix is a document used for tracking the requirement, test cases and the defects
- This document is made to convince the client that the coverage done is complete as end to end
- This document contains Requirement id, test case condition and bug id
- Using this document the person can track the requirement based on bug id

High Level vs Low Level Test Case

High Level Test Case	Low Level Test Case
High Level Test cases cover the whole application in a broader way. They do not cover the functionalities in detail but the overall application should work fine	Low level test cases cover each and every individual units of code and test cases are generated to test each and every unit in depth
Abstract	More Detailed

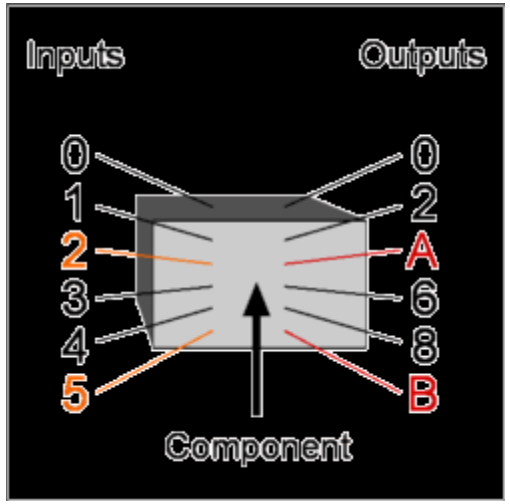
Testing Types

- Black Box Testing
- White Box Testing
- Gray Box Testing

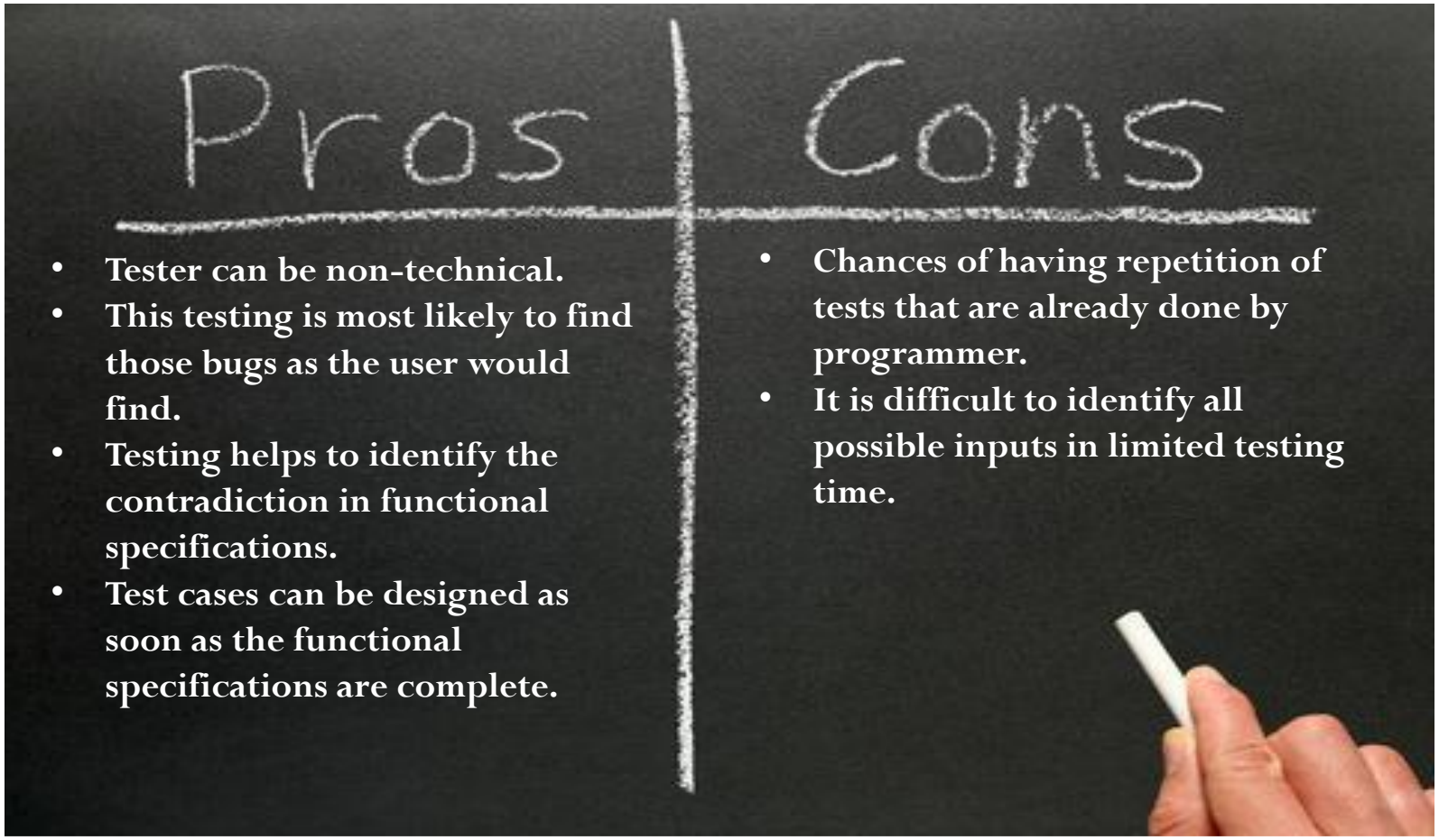


Black Box Testing

- In science and engineering, a **black box** is a device, system or object which can be viewed solely in terms of its input, output and transfer characteristics without any knowledge of its internal workings, that is, its implementation is "opaque" (black).
- Also known as functional testing. A software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on a software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications.

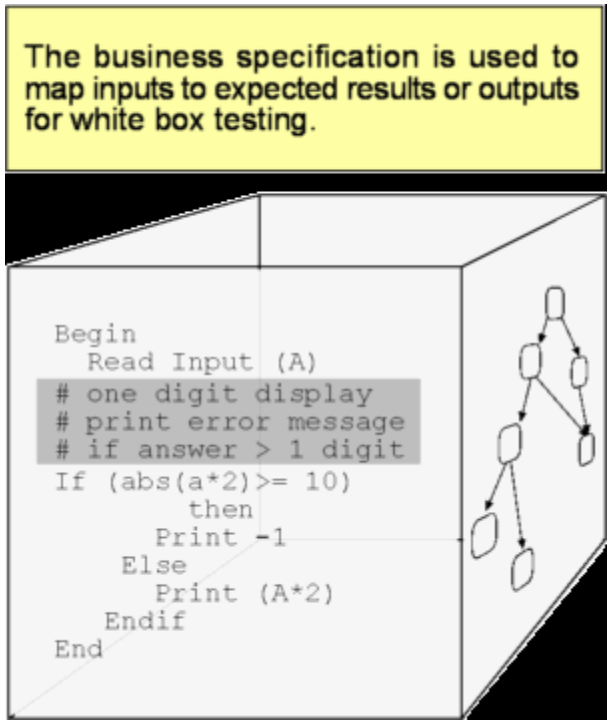


Black Box Testing

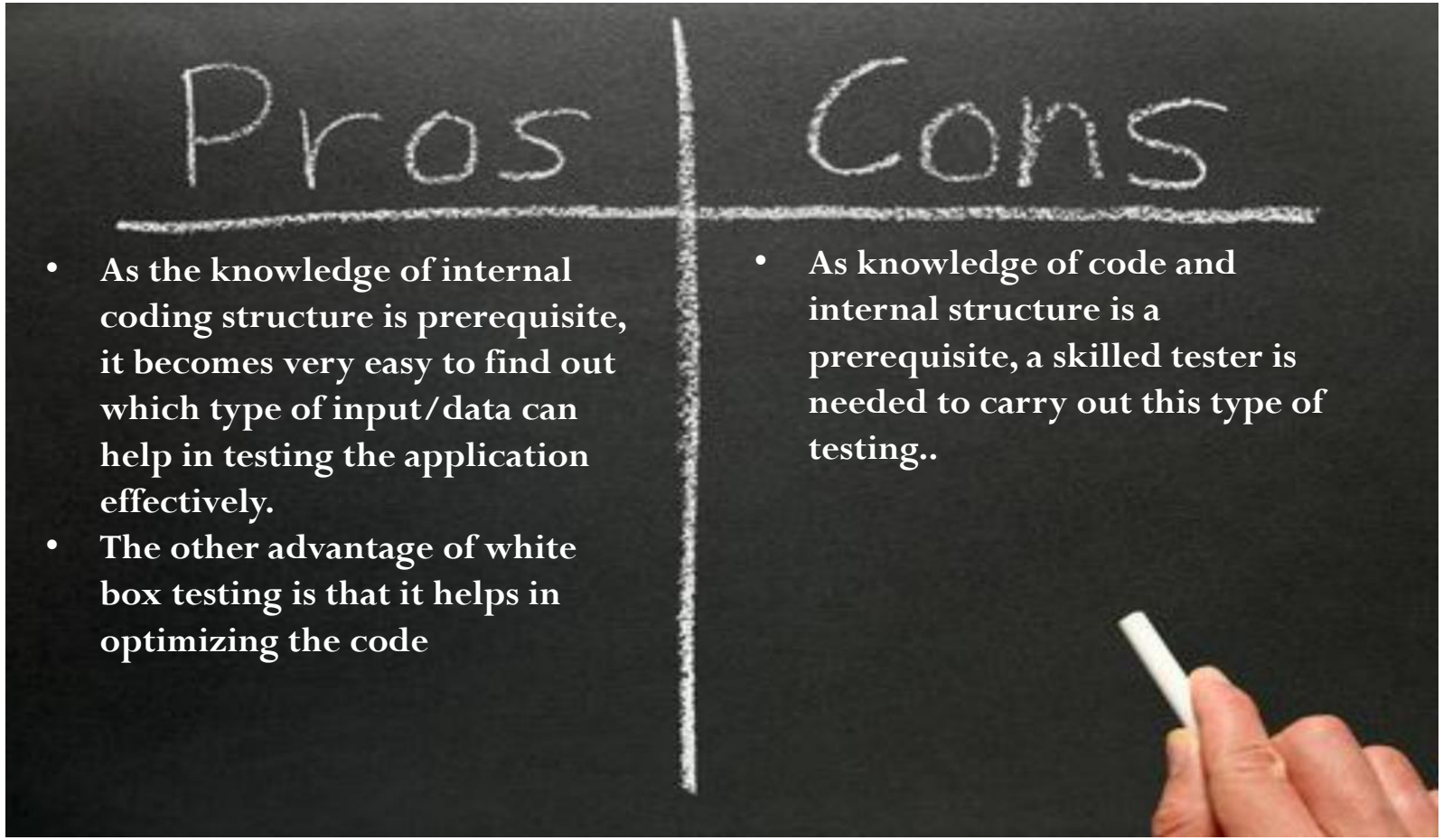


White-Box Testing

- **White-box testing** (also known as **clear box testing**, **glass box testing**, **transparent box testing**, and **structural testing**) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).
- The connotations of "clear box" and "glass box" appropriately indicate that you have full visibility of the internal workings of the software product, specifically, the logic and the structure of the code.
- In white-box testing an internal perspective of the system, as well as programming skills, are required and used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.



White-Box Testing



Gray Box Testing

- Gray box testing is a software testing technique that uses a combination of black box testing and white box testing. Gray box testing is not black box testing, because the tester does know some of the internal workings of the software under test.
- In gray box testing, the tester applies a limited number of test cases to the internal workings of the software under test. In the remaining part of the gray box testing, one takes a black box approach in applying inputs to the software under test and observing the outputs.
- This is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

References

- Software Engineering by Roger Pressman
- http://www.faqs.org/faqs/software-eng/testing-faq/section-14.html
- http://en.wikipedia.org/wiki/White-box_testing
- http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf
- http://blogs.ebusinessware.com/2009/06/26/unit-testing-vs-module-testing/
- John Watkins ,"Testing IT", 2001, Cambridge University Press
- GlenFord Myers, "The Art of Software Testing" 2nd Edition
- http://www.onestoptesting.com
- http://www.softwaretestingmentor.com/automation/manual-vs-automation.php