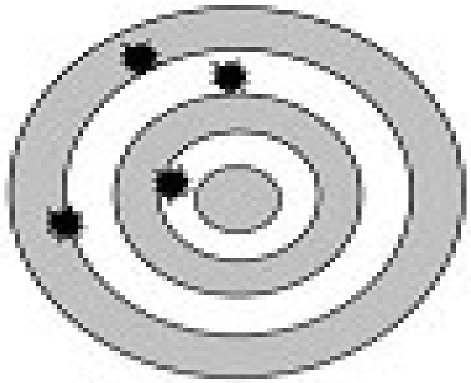
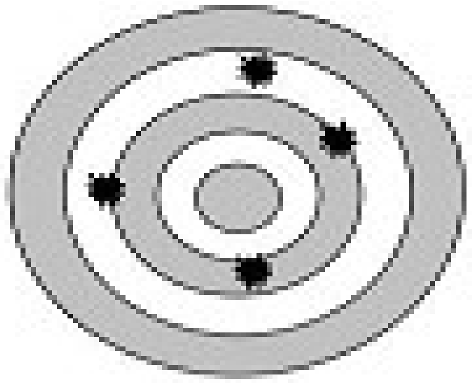
Software Testing – Terms & Def^{n}

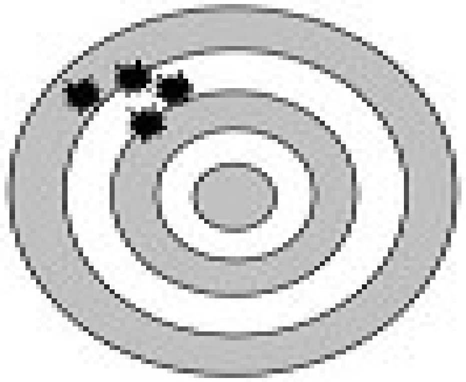
- Precision and Accuracy

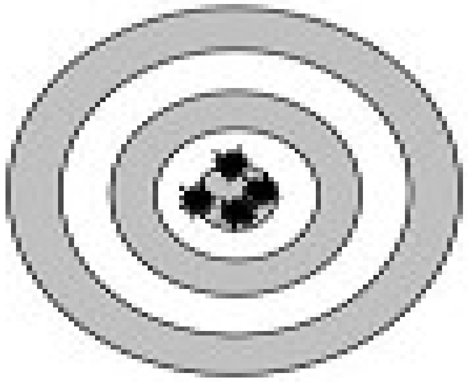
  ex – Calculator – Should you test that answers are precise or accurate

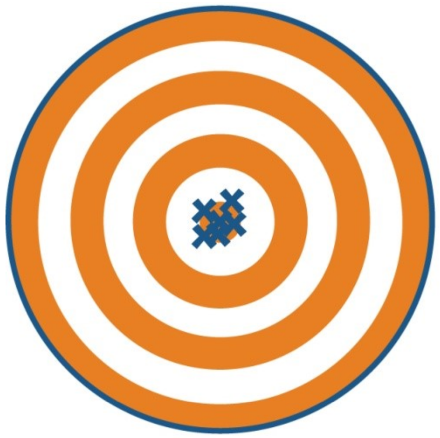
Precision and Accuracy

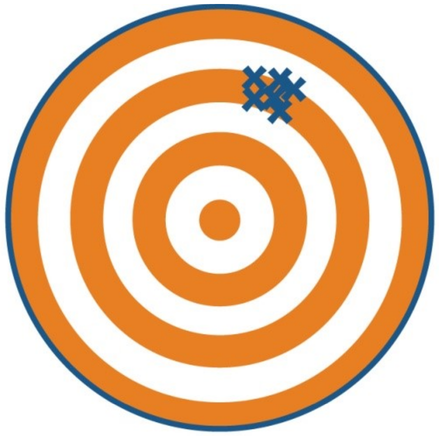

Not Accurate
Not Precise

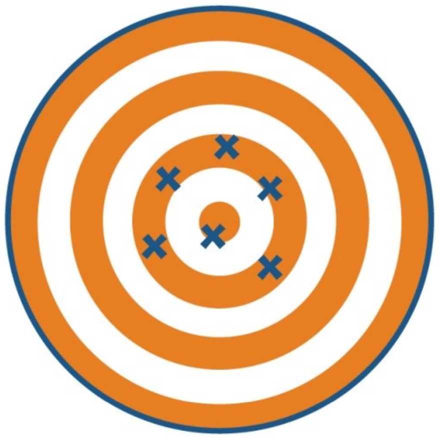

Accurate
Not Precise

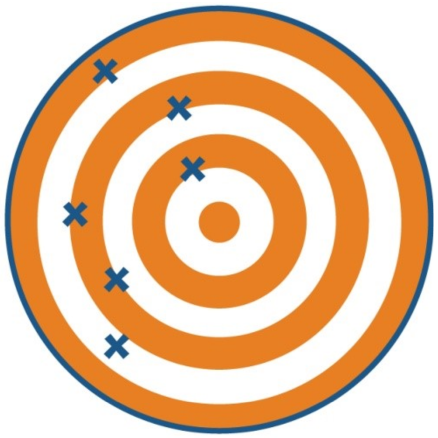

Not Accurate
Precise


Accurate
Precise


High Accuracy
High Precision


Low Accuracy
High Precision


High Accuracy
Low Precision


Low Accuracy
Low Precision

The Accuracy and precision depends on what the product is and ultimately what the development team is aiming at.

Verification and Validation

"I landed on "Go" but didn't get my $200!"

Are we building the product right?

"I know this game has money and players and "Go" – but this is not the game I wanted."

Are we building the right product?

Software Testing – Terms & Def^{n}

- Verification and Validation –

Verification is the process confirming that something -*(Software)*- meets its specification.

Are we building the product right?

Validation is the process confirming that it meets the user's requirements. **Process designed to uncover problems, increase confidence.** Combination of reasoning and test

Are we building the right product?

Example – Mirror of Hubble Telescope.

Mirror was precise but not accurate.

Software Testing – Terms & Def^{n}

- Quality and Reliability

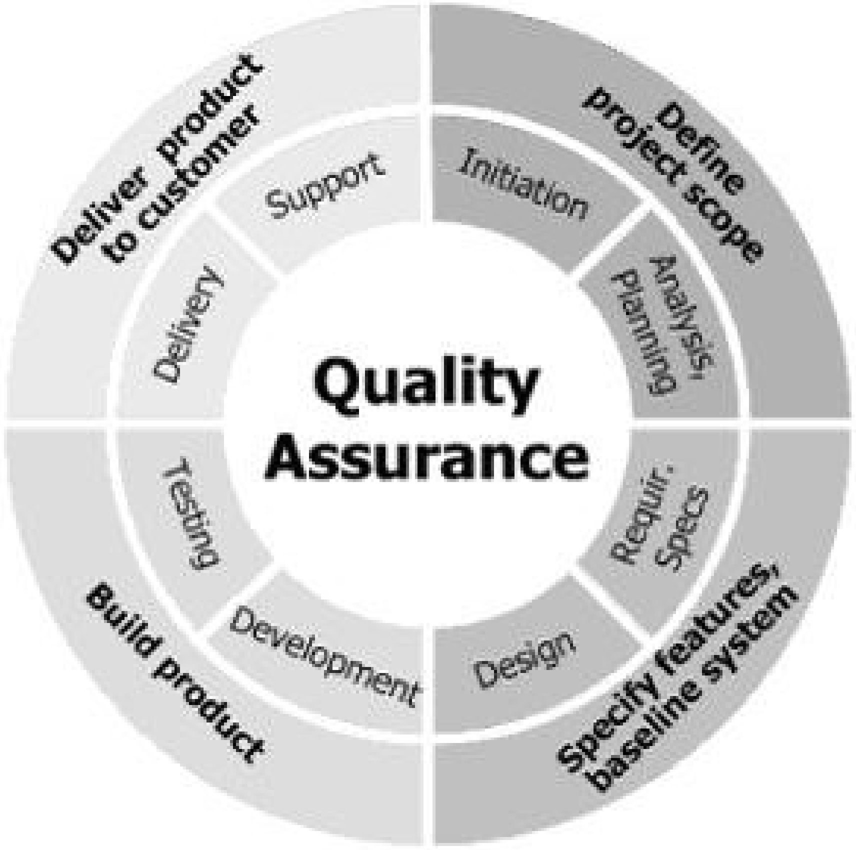
  Quality – defined as " *a degree of excellence*" or "Superiority in kind"

  If the S/W is of high quality it will meet the customers needs.

  Software reliability is the most important and most measurable aspect of software quality and it is very customer oriented. It is a measure of how well the program functions to meet its operational requirements.

- Software reliability is defined as the probability of failure-free operation of a computer program for a specified time in a specified environment. For example, a program might have a reliability of 0.82 for 8 hours of execution. A failure is a departure of program operation from requirements.

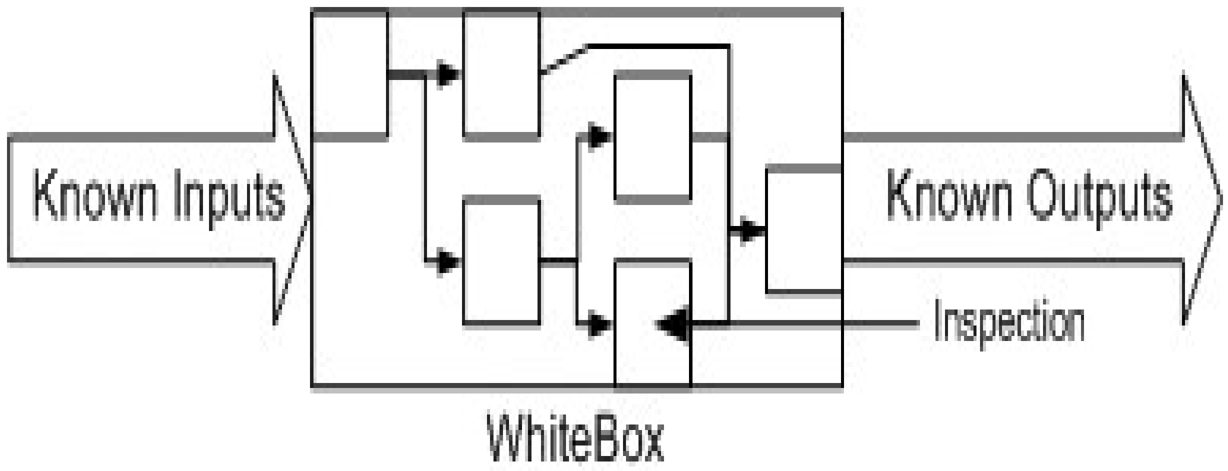
Testing and Quality Assurance



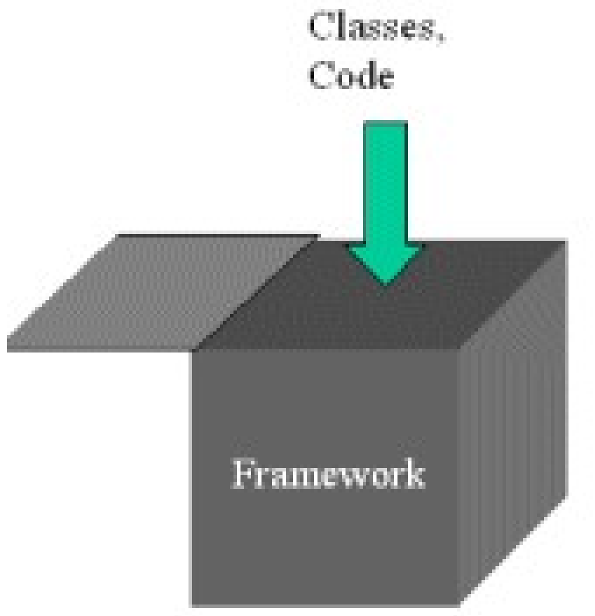
- Responsibility of S/W tester – Find bugs, find them as early as possible, and make sure they get fixed.
- Responsibility of Quality Assurance person – to create and enforce standards and methods to improve the development process and to prevent bugs from ever occurring.

White Box testing

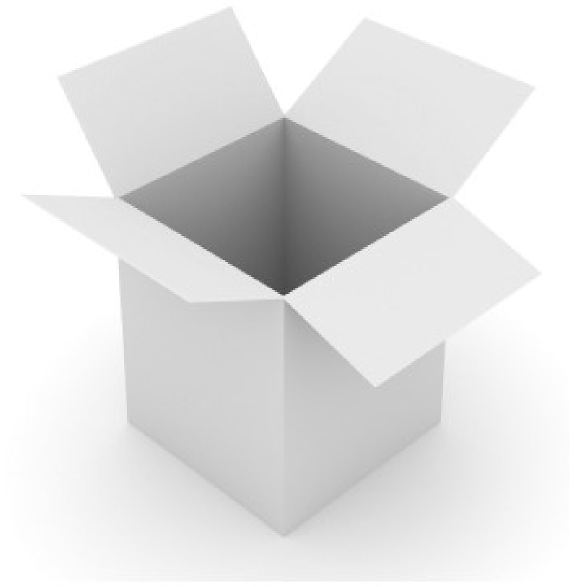
- White Box Testing / Clear box testing)



White Box testing



White-box framework



White Box testing

- White-box testing is a verification technique software engineers can use to examine if their code works as expected.
- The S/W tester has the access to the programmer’s code and can examine it for clues to help him with his testing

White Box testing

- White-box testing is also known as *structural testing*, *clear box testing,* and *glass 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.

White Box testing

Using the white-box testing techniques, a software engineer can design test cases that

(1) exercise independent paths within a module or unit;

(2) Exercise logical decisions on both their true and false side;

(3) execute loops at their boundaries and within their operational bounds; and

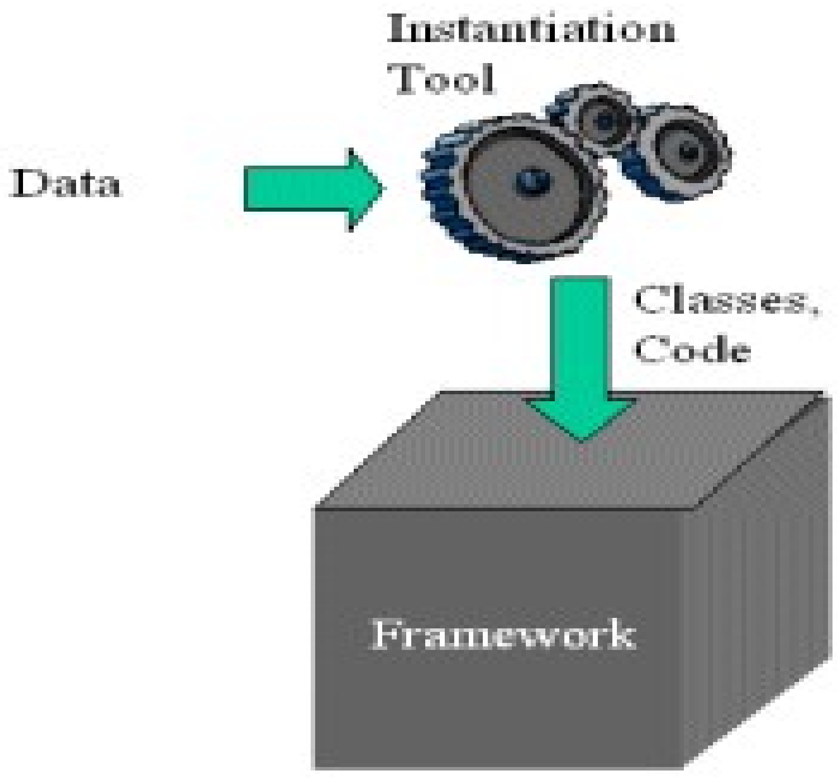
(4) exercise internal data structures to ensure their validity

- Look at the code (white-box) and try to systematically cause it to fail

Coverage criteria: a way to be systematic

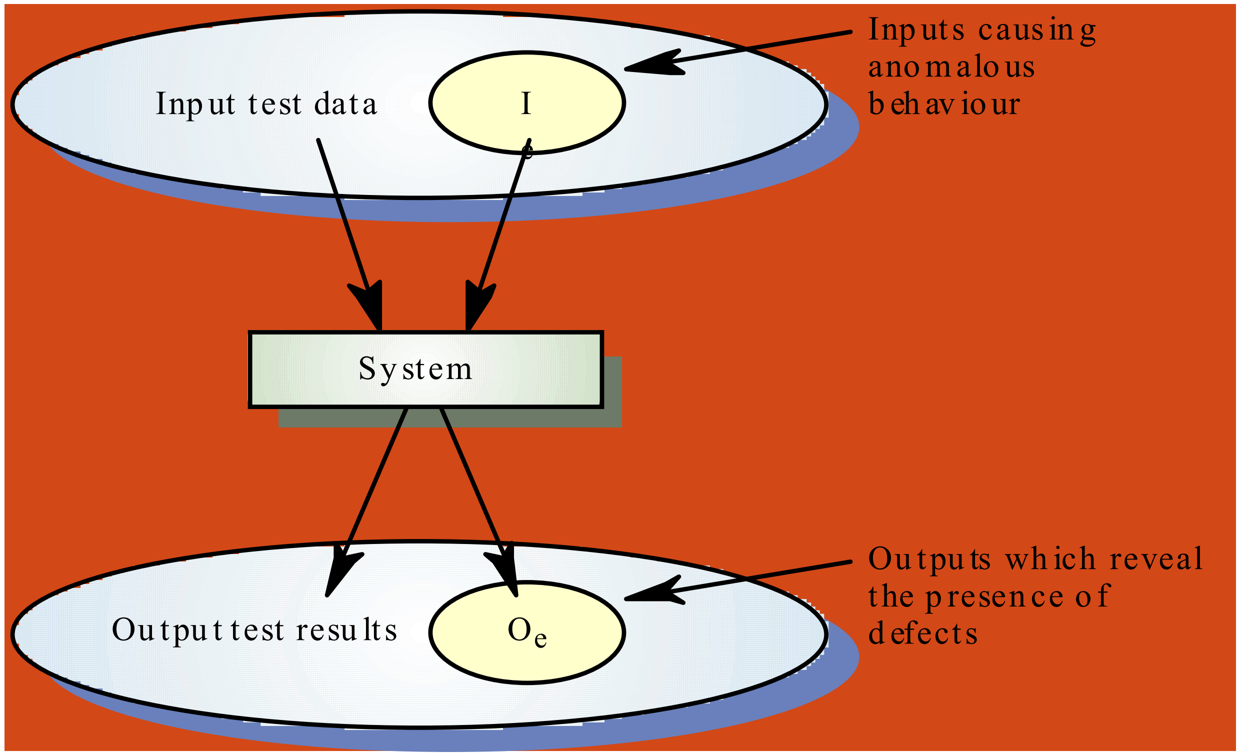
- Function coverage
    - Execute each function
- Statement coverage
    - Most common
- Edge coverage
    - Take both sides of each branch
- Path coverage
    - Note: infinite number of paths!
    - Typical compromise: 0-1-many loop iterations
- Condition coverage
    - Choose a set of predicates
    - Cover each statement with each combination of predicates
- Exercise data structures
    - Each conceptual state or sequence of states
- Typically cannot reach 100% coverage

Black Box testing



Black-box framework

Black-box testing


Inputs causing anomalous behaviour
Input test data
I_e
System
Output test results
O_e
Outputs which reveal the presence of defects

Black Box testing

- Black-box test design is usually described as focusing on testing functional requirements. Synonyms for black-box include: behavioral, functional, opaque-box, and closed-box.
- The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure

Black Box testing

- Verify each piece of functionality of the system

  Black-box: don't look at the code
- Systematic testing
  - Test each use case
  - Test combinations of functionality (bold + italic + font + size)
    - Generally have to sample
  - Test incorrect user input
  - Test each "equivalence class" (similar input/output)
  - Test uncommon cases
    - Generating all error messages
    - Using uncommon functionality
  - Test borderline cases
    - Edges of ranges, overflow inputs, array of size 0 or 1

Black Box testing

- Black box testing attempts to find errors in the external behavior of the code in the following categories

(1) incorrect or missing functionality;
(2) interface errors;
(3) errors in data structures used by interfaces;
(4) behavior or performance errors;
(5) initialization and termination errors.