WHITE BOX TESTING

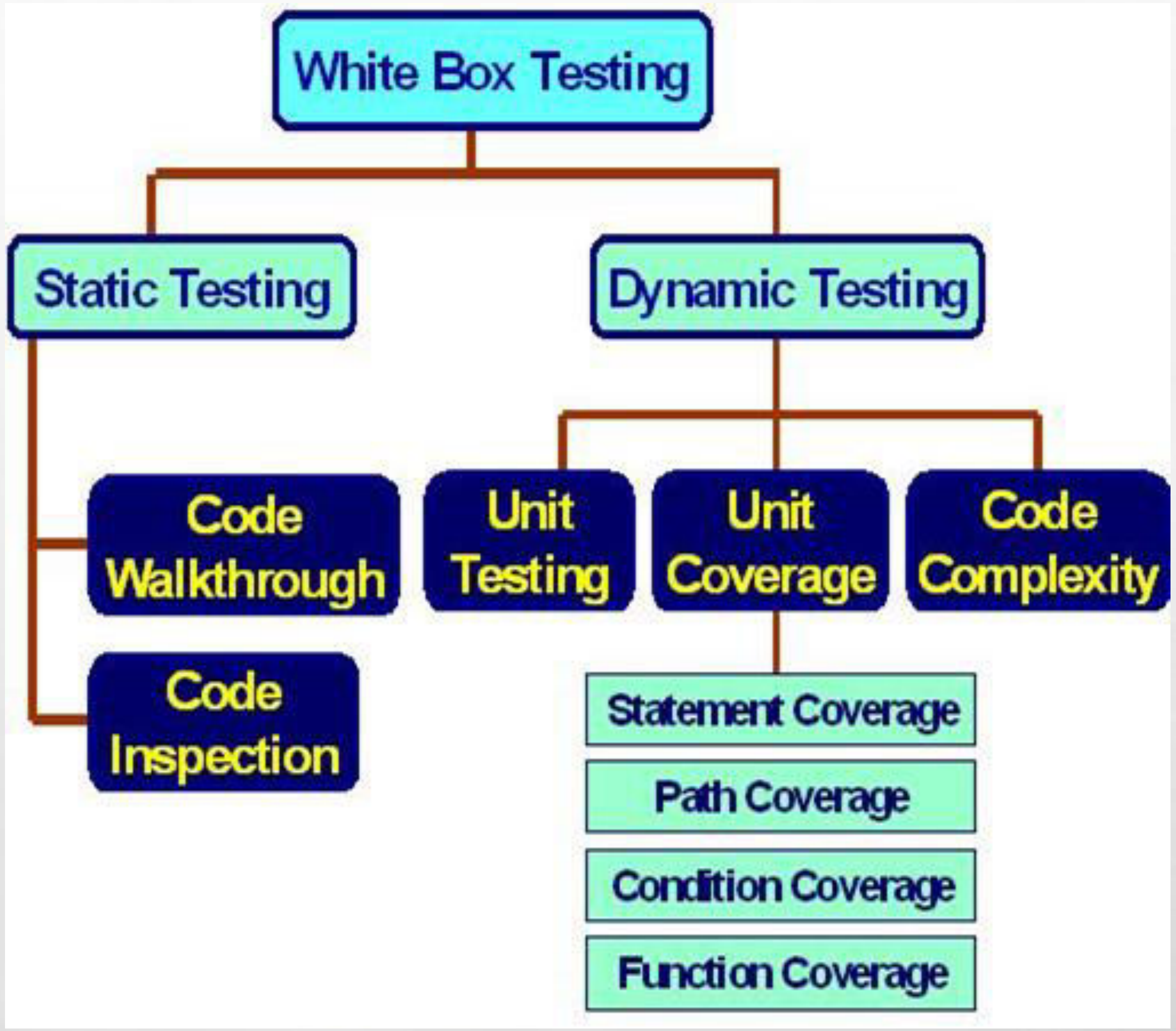
WHITE BOX TESTING

- WHITE BOX TESTING IS A WAY OF TESTING THE EXTERNAL FUNCTIONALITY OF THE CODE BY EXAMINING AND TESTING THE PROGRAM CODE THAT REALIZES THE EXTERNAL FUNCTIONALITY.
- THIS IS ALSO KNOWN AS CLEAR BOX, OR GLASS BOX OR OPEN BOX TESTING.
- WHITE BOX TESTING TAKES INTO ACCOUNT THE PROGRAM CODE, CODE STRUCTURE, AND INTERNAL DESIGN FLOW.

What do you verify in White Box Testing?

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

CLASSIFICATION OF WHITE BOX TESTING



STATIC TESTING

- IT REQUIRES ONLY THE SOURCE CODE OF THE PRODUCT, NOT THE BINARIES OR EXECUTABLES.
- STATIC TESTING DOES NOT INVOLVE EXECUTING THE PROGRAMS ON COMPUTERS BUT INVOLVES SELECT PEOPLE GOING THROUGH THE CODE TO FIND OUT WHETHER
  - ✓ THE CODE WORKS ACCORDING TO THE FUNCTIONAL REQUIREMENT;
  - ✓ THE CODE HAS BEEN WRITTEN IN ACCORDANCE WITH THE DESIGN DEVELOPED EARLIER IN THE PROJECT LIFE CYCLE;
  - ✓ THE CODE FOR ANY FUNCTIONALITY HAS BEEN MISSED OUT;
  - ✓ THE CODE HANDLES ERRORS PROPERLY.

TYPES OF STATIC TESTING

- DESK CHECKING
- CODE WALKTHROUGH
- CODE INSPECTION

Desk Checking

- NORMALLY DONE MANUALLY BY THE AUTHOR OF THE CODE.
- IT IS A METHOD TO VERIFY THE PORTIONS OF THE CODE FOR CORRECTNESS.
- MOST PROGRAMMERS DO DESK CHECKING BEFORE COMPILING AND EXECUTING THE CODE.
- WHENEVER ERRORS ARE FOUND, THE AUTHOR APPLIES THE CORRECTIONS FOR ERRORS ON THE SPOT.

Code Walkthrough

- WALKTHROUGH IS GROUP-ORIENTED METHOD AND ARE LESS FORMAL THAN INSPECTIONS.
- IN WALKTHROUGHS, A SET OF PEOPLE LOOK AT THE PROGRAM CODE AND RAISE QUESTIONS FOR THE AUTHOR.
- THE AUTHOR EXPLAINS THE LOGIC OF THE CODE, AND ANSWERS THE QUESTIONS.
- IF THE AUTHOR IS UNABLE TO ANSWER SOME QUESTIONS, HE OR SHE THEN TAKES THOSE QUESTIONS AND FINDS THEIR ANSWERS.

Code Inspection

- IT IS ALSO CALLED FAGAN INSPECTION, NORMALLY TAKES PLACE WITH A HIGH DEGREE OF FORMALISM.
- THE FOCUS OF THIS METHOD IS TO DETECT ALL FAULTS, VIOLATIONS, AND OTHER SIDE-EFFECTS.
- A FORMAL INSPECTION SHOULD TAKE PLACE ONLY WHEN THE AUTHOR HAS MADE SURE THE CODE IS READY FOR INSPECTION BY PERFORMING SOME BASIC DESK CHECKING AND WALKTHROUGHS.

STRUCTURAL TESTING

- STRUCTURAL TESTING TAKES INTO ACCOUNT THE CODE, CODE STRUCTURE, INTERNAL DESIGN, AND HOW THEY ARE CODED.

- THE FUNDAMENTAL DIFFERENCE BETWEEN STRUCTURAL TESTING AND STATIC TESTING IS THAT IN STRUCTURAL TESTING TESTS ARE ACTUALLY RUN BY THE COMPUTER ON THE BUILT PRODUCT, WHEREAS IN STATIC TESTING, THE PRODUCT IS TESTED BY HUMANS

TYPES OF STRUCTURAL TESTING

- STATEMENT COVERAGE
- PATH COVERAGE
- CONDITION COVERAGE
- FUNCTION COVERAGE

Statement Coverage

- PROGRAM CONSTRUCTS IN MOST CONVENTIONAL PROGRAMMING LANGUAGES CAN BE CLASSIFIED AS
  - SEQUENTIAL CONTROL FLOW
  - TWO-WAY DECISION STATEMENTS LIKE IF THEN ELSE
  - MULTI-WAY DECISION STATEMENTS LIKE SWITCH
  - LOOPS LIKE WHILE DO, REPEAT UNTIL AND FOR
- STATEMENT COVERAGE REFERS TO WRITING TEST CASES THAT EXECUTE EACH OF THE PROGRAM STATEMENTS.

Statement Coverage

- STATEMENT COVERAGE=(TOTAL STATEMENTS EXERCISED / TOTAL NUMBER OF EXECUTABLE STATEMENTS IN PROGRAM) * 100

Statement Coverage

```
INT F (INT X, INT Y)
    WHILE (X!=Y)  {
        IF(X>Y) THEN
            X=X-Y;
        ELSE Y=Y-X;
    }
    RETURN X; }
```

Statement Coverage

```
INT F (INT X, INT Y)
    WHILE (X!=Y)  {
        IF(X>Y) THEN
            X=X-Y;
        ELSE Y=Y-X;
    }
    RETURN X; }
```

{ (X = 3, Y = 3)

(X = 4, Y = 3)

(X = 3, Y = 4) }

Path Coverage

- IN PATH COVERAGE, WE SPLIT A PROGRAM INTO A NUMBER OF DISTINCT PATHS. A PROGRAM (OR A PART OF A PROGRAM) CAN START FROM THE BEGINNING AND TAKE ANY OF THE PATHS TO ITS COMPLETION.

- PATH COVERAGE = (TOTAL PATHS EXERCISED / TOTAL NUMBER OF PATHS IN PROGRAM) * 100

Condition Coverage

- IT IS NECESSARY TO HAVE TEST CASES THAT EXERCISE EACH BOOLEAN EXPRESSION AND HAVE TEST CASES TEST PRODUCE THE TRUE AS WELL AS FALSE PATHS.
- OBVIOUSLY, THIS WILL MEAN MORE TEST CASES AND THE NUMBER OF TEST CASES WILL RISE EXPONENTIALLY WITH THE NUMBER OF CONDITIONS AND BOOLEAN EXPRESSIONS.
- CONDITION COVERAGE = (TOTAL DECISIONS EXERCISED / TOTAL NUMBER OF DECISIONS IN PROGRAM) * 100

Condition Coverage

```
INT F (INT X, INT Y)
    WHILE (X!=Y)  {
        IF(X>Y) THEN
            X=X-Y;
        ELSE Y=Y-X;
    }
    RETURN X; }
```

{ (X = 3, Y = 3)

(X = 3, Y = 2)

(X = 4, Y = 3)

(X = 3, Y = 4) }

Function Coverage

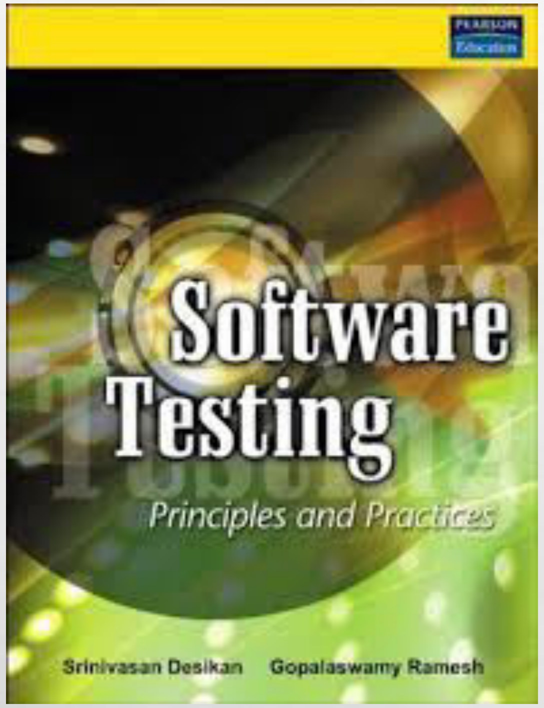
- IDENTIFY HOW MANY PROGRAM FUNCTIONS (SIMILAR TO FUNCTIONS IN “C” LANGUAGE) ARE COVERED BY TEST CASES.

- FUNCTION COVERAGE = (TOTAL FUNCTIONS EXERCISED / TOTAL NUMBER OF FUNCTIONS IN PROGRAM) * 100

Function Coverage Advantage

- FUNCTIONS ARE EASIER TO IDENTIFY IN A PROGRAM AND HENCE IT IS EASIER TO WRITE TEST CASES TO PROVIDE FUNCTION COVERAGE
- SINCE FUNCTIONS ARE AT A MUCH HIGHER LEVEL OF ABSTRACTION THAN CODE, IT IS EASIER TO ACHIEVE 100 PERCENT FUNCTION COVERAGE.
- FUNCTIONS HAVE A MORE LOGICAL MAPPING TO REQUIREMENTS AND HENCE CAN PROVIDE A MORE DIRECT CORRELATION TO THE TEST COVERAGE OF THE PRODUCT.
- SINCE FUNCTIONS ARE A MEANS OF REALIZING REQUIREMENTS, THE IMPORTANCE OF FUNCTIONS CAN BE PRIORITIZED BASED ON THE IMPORTANCE OF THE REQUIREMENTS THEY REALIZE.

REFERENCES



Software Testing

Principles and Practices

Srinivasan Desikan Gopalaswamy Ramesh

Guru99

edureka!

Software Testing Help