CODE COVERAGE FUNDAMENTALS

What is code coverage?

Is a measure which describes the degree of which the source code of the program has been tested.

Why use Code Coverage?

- It helps you to measure the efficiency of test implementation
- It offers a quantitive measurement
- It defines the degree to which the source code has been tested

Code Coverage Methods

- **Statement Coverage**
  - Is a white box test design technique, which involves execution of all the executable statements in the source code at least once.
  - SC = 100*(Number of executed statements)/(Total number of statements)
- **Decision Coverage**
  - Reports the true or false outcomes of each Boolean expression.
  - DC = Number of Decision Outcomes Exercised / Total Number of Decision Outcomes
- **Branch Coverage**
  - In the branch coverage, every outcome from a code module is tested.
  - Branch Coverage = Number of Executed Branches / Total Number of Branches
  - Advantages
    - Allows you to validate-all the branches in the code
    - Ignores branches inside the Boolean expressions
- **Condition Coverage**
  - CC will reveal how the variables/subexpressions in the conditional statement are evaluated.
  - CC = Number of Executed Operands / Total Number of Operands
- **Finite State Machine (FSM) Coverage**
  - It works on the behaviour of the design
  - It checks how many sequences are included in a finite state machine.

Which type of Code Coverage to Choose

Tester needs to check that the

- Code under test has single or multiple undiscovered defects
- Cost of the potential penalty
- Cost of lost reputation
- Cost of lost sale, etc.

The higher the probability of defects, the more severe the level of coverage you need to choose

Advantages of Using Code Coverage

- Helpful to evaluate a quantitative measure of code coverage
- It allows you to create extra test cases to increase coverage
- It allows you to find the areas of a program which is not exercised by a set of test cases

Disadvantages of Using Code Coverage

- Even when any specific feature is not implemented in design, code coverage still report 100% coverage.
- It is not possible to determine whether we tested all possible values of a feature with the help of code coverage.

HOW TO MISUSE CODE COVERAGE

Using code coverage well

- Think of what feature in the interface that condition corresponds to.
- Rethink how I should have tested that before
- Run your new tests and recheck coverage.
- Repeat for other features.

What's all this about "having enough time"?

- Divide the code under test into 3 categories
  - **High risk** code could cause severe damage, or has many users, or seems likely to have many mistakes whose costs will add up.
  - **Low risk** code is unlikely to have bugs important enough to stop or delay a shipment, even when all the bugs are summed together.
  - **Medium risk** code is somewhere in between. Bugs here would not be individually critical, but having too many of them would cause a schedule slip.
- Test the high risk code thoroughly . Use most of the remaining time testing the medium risk code.
- Check coverage
  - Since high risk code is tested, I expect good coverage.
  - I expect lower coverage for medium risk code.
  - The coverage for low risk code is pretty uninteresting.

How product testers can misuse code coverage

1.- Examine the tests and compare them to the external description of what that part of the product is to do

2-. Are the tests really testing what you thought they were testing?

3.- Talk briefly to the developer to get an idea of what you missed.

- It's not cost effective. You don't have time to learn the code
- You should be as worried about discovering errors the code overlooks as about exercising the errors it purpots to handle.

Product testers have more tasks than developer testers.

1) They either supplement or replace developer testing by testing particular features in isolation.
2) They also test user scenarios that span features
3) Product testers also test the product against reasonable user expectations.