Code Coverage & Tools

www.scmGalaxy.com

Author: Rajesh Kumar
rajesh@scmGalaxy.com

scmGalaxy

What is Code Coverage

- Code Coverage is an important measurement in Software Quality Engineering. While Software testing ensures correctness of the applications, a metric is required to track the completeness and effectiveness of the testing undertaken.
- Code Coverage helps achieve reliable quality through identifying untested areas of the application.

Why Code Coverage

Software testing is a challenging function. The testers need to ensure complete functional and non-functional correctness of the product. Considering the complex workflows and use cases of modern day applications, the number of unique cases that the software can be used often run into millions, which is not feasible to be covered under testing exercise. The testers thus need to

- While Planning Tests

o Ensure covering all workflows in terms of decision trees in the code

o Ensure covering all data values – by identifying patterns rather covering millions of values

- While testing

o Ensuring the testing is completely exercising the whole application with planned and exploratory tests.

Why Code Coverage cont...

At the end of testing, the decision to stop testing and release the product still remains subjective, based on the presence or absence of bugs, inflow of new bugs, success rate of each test cycle, confidence rating of the testers or users, etc. Whereas the definitive metric of quantifying how much of the application was really tested, is missed.

Code Coverage is measured as quantification of application code exercised by the testing activities. Code Coverage can be measured at various levels – in terms of programming language constructs – Packages, Classes, Methods, Branches or in terms of physical artifacts - Folders, Files and Lines. For Eg. A Line Coverage metric of 67% means the testing exercised 67% of all executable statements of the application. A Code Coverage metric usually is accompanied by Code Coverage Analysis Report – which helps identify the un-tested part of the application code, thereby giving the testers early inputs for complete testing.

Benefits of Code Coverage

- Objective Indicator of Test Coverage of application code
- Pointers to uncovered Packages / Classes / Methods / Branches
- Pointers to uncovered Folders / Files / Lines
- Drill down to untested part of source code and devise new tests
- Early Indicator for Testing Quality and Fixing it by adding new tests.
- Remove redundancy in testing
- Increased Confidence for Releases

Test Your Test

Typical Emotional Storyboard

- Write Some code! Happy!
- Does it work? Sad!
- Write some test! Happy!
- Do they really test the code? Sad!
- Measure the Code Coverage! Happy!

Coverage Measurement

- Shows Which line of code are executed
- How much of your code is covered by your tests?
- Your tests test your product
- Coverage testing tests your tests

Goal

- **100%**
  - Coverage Ideal
  - Not Always possible
  - Can be expensive to achieve
  - Design for testability

Practical Goal

- More Coverage is better: Good
- Actual number does not matter
- False quantifiability: bad
- Use coverage results to understand your code

Good: Write more tests

Only way to truly increase code coverage

Bad

Excluding Code to boost Coverage

Types of Coverage

- Statement Coverage
- Branch Coverage
- Path Coverage
- Loop Path Coverage
- Data – driven Code
- Complex Conditionals
- Hidden Branches

Code Coverage Terminologies

Instrumentation

Instrumentation is the process to add code to the application to be able to output Code Coverage data. Instrumentation can be done at Source Level or at Intermediate language level, and rarely at run-time as information collection.

Source Level Instrumentation: Prior to compilation, instrumentation code are inserted by the code coverage tool to the application. This will be compiled into the application code.

Object level Instrumentation: Post compilation of the Application, executable lines to collect Coverage data are injected into the Object code.

Merge

Ability to run tests in batches, or in different environments, yet consolidate the overall coverage reports. Most of good coverage tools support offline merge feature.

Tooling Infrastructure

- For Code Coverage analysis the ideal infrastructure would be
- - Integrated Instrumentation with the build process
- - Deployment of Code Coverage instrumented application
- - Auto-Collection of Coverage data during testing
- - Merge & Final Report automation
- Code Coverage reports can be generated on every test cycle if tests.

Code Coverage is NOT

- What Code Coverage is and is Not100% Code Coverage Does Not say the product is bug free, it ensures product is 100% tested. If the product was tested wrongly, Code Coverage can't help.
- Code Coverage testing does not require any special test methods – On the instrumented build – any testing that is carried out will generate coverage data. Regular using of the instrumented application will generate coverage information as well.
- Code Coverage is not about Whitebox testing. Code coverage is generated when you test the application in Any way. To analyse the code, the review happens at Code Level. It is not Whitebox testing.
- Code Coverage is not only through Unit Testing or Automated Testing.
- Code Coverage does not require extra testing efforts
- Code Coverage is not an end game activity, the earlier the better. It gives scope to improve tests and cover more code, thereby ensuring higher quality.
- Code Coverage instrumented builds can't be used for Performance Testing. It will add performance overhead.

How?

Coverage Tools

- Clover
- Cobertura
- Emma

Thank You !

www.scmGalaxy.com

Author: Rajesh Kumar
rajesh@scmGalaxy.com