Unit, Integration & Functional Testing

Tejas Parikh (t.parikh@northeastern.edu)
Spring 2018

CSYE 6225
Northeastern University
https://spring2018.csye6225.com

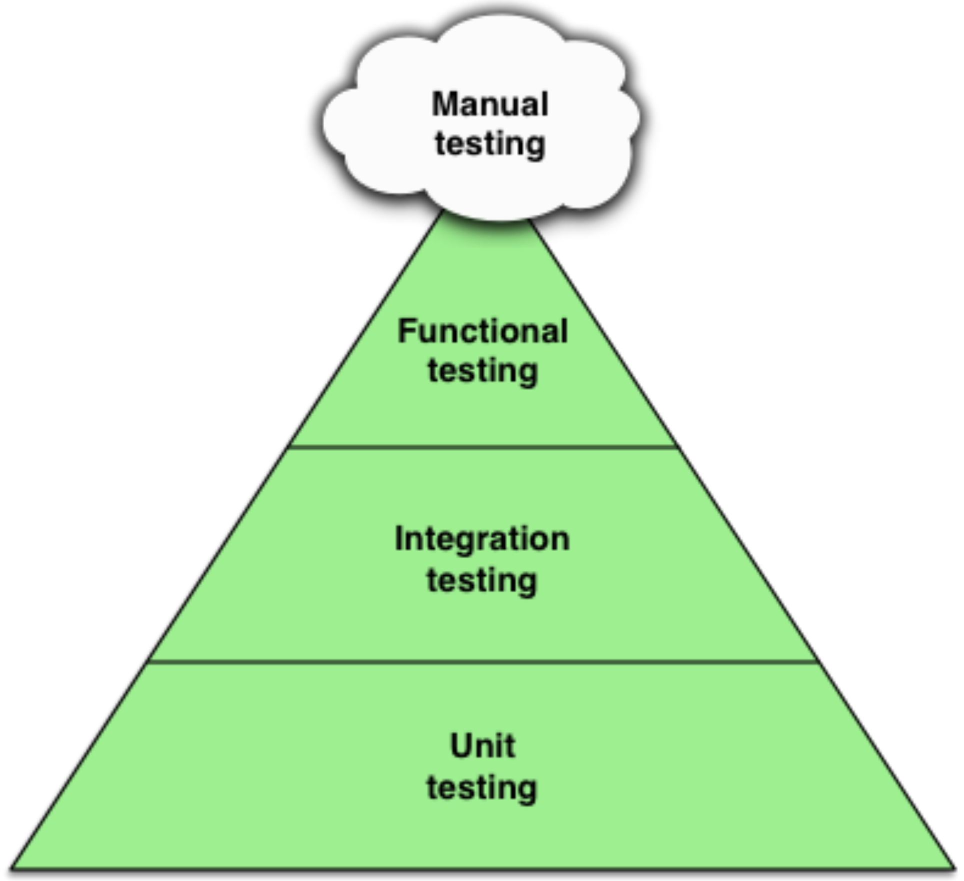
I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO I DO IT IN PRODUCTION
memegenerator.net

Why Test?

- Testing reduces bugs
- Tests Service as good documentation
- Tests allow for safe refactoring
- Tests reduce the cost of making code changes
- Tests allow you to deliver code with confidence

Types of Tests

- There are way too many testing level and types to list them here. We will focus on Unit and Integration tests.



What is Unit Test?

- Unit tests should not require access to any external systems such as network, databases, etc.
- All external systems such as database, file server, etc. are mocked out using specific test APIs and test data.

Unit Tests

- Isolate parts of programs
- Verify that independent part of programs are working correctly
- Unit tests are fast & reliable
- However, Unit tests
  - ➢Take time to build
  - ➢Require maintenance
- Both of these points require significant time and commitment. An incorrect unit test can let bug go thru unnoticed for long time.

I have an ongoing fear that all my unit tests pass, but they're meaningless.

What is Integration Test?

- Integration tests verify that interaction between multiple components (applications, services, modules, etc.) is working as expected.

Integration Test Challenges

- Difficult to test all critical paths
- Hard to find the source of errors
- Requires time and commitment from multiple component owners

Performance/Load/Stress Testing

- Simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types.
- Load testing is also a way to perform a functional test on websites, databases, LDAPs, webservices etc.

Test Impact Analysis

- Test Impact Analysis (TIA) is a modern way of speeding up the test automation phase of a build. It works by analyzing the call-graph of the source code to work out which tests should be run after a change to production code.

Why Test Impact Analysis?

- "Too Many" tests to run prior to check-in
- Developers may ignore tests if it takes too long to run.
- Test Impact Analysis (TIA) is a technique that helps determine which subset of tests for a given set of changes.

Continuous Integration & Delivery

Continuous Integration

- **Continuous integration** (**CI**) is the practice of merging all developer working copies to a shared mainline several times a day.
- The main aim of CI is to prevent integration problems, referred to as "integration hell".
- CI is intended to be used in combination with automated unit tests written through the practices of test-driven development.
- In addition to automated unit tests, organizations using CI typically use a build server to implement continuous processes of applying quality control in general — small pieces of effort, applied frequently. In addition to running the unit and integration tests, such processes run additional static and dynamic tests, measure and profile performance, extract and format documentation from the source code and facilitate manual QA processes.
- This continuous application of quality control aims to improve the quality of software, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control *after* completing all development.

CI – Best Practices

- Continuous integration – the practice of frequently integrating one's new or changed code with the existing code repository – should occur frequently enough that no intervening window remains between commit and build, and such that no errors can arise without developers noticing them and correcting them immediately.
- Normal practice is to trigger these builds by every commit to a repository, rather than a periodically scheduled build.
- The practicalities of doing this in a multi-developer environment of rapid commits are such that it is usual to trigger a short time after each commit, then to start a build when either this timer expires, or after a rather longer interval since the last build.
- Many automated tools offer this scheduling automatically.

Continuous Integration Principles

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

CI Benefits

- Bugs are detected early
- Avoids last-minute chaos at release dates
- When unit tests fail or a bug emerges, if developers need to revert the codebase to a bug-free state without debugging, only a small number of changes are lost (because integration happens frequently).
- Constant availability of a "current" build for testing, demo, or release purposes
- Frequent code check-in pushes developers to create modular, less complex code

Continuous Deployment (CD)

- Continuous deployment is another core concept in a DevOps strategy.
- CD's primary goal is to enable automated deployment of production-ready application code.
- Continuous deployment is sometimes referred to as Continuous Delivery.
- By using continuous delivery practices and tools, software can be deployed rapidly, repeatedly, and reliably. If deployment fails, it can be automatically rolled back to previous version.

Additional Resources

https://spring2018.csye6225.com