Software quality and testing

- 1st course

Ádám István Szűcs | Budapest | February, 2018

Contents of Today

- Introduction, Agenda & Syllabus of the semester
- Tools:
  - **JetBrains IntelliJ IDEA**
- Defining a Unit test
- Contrasting unit testing with integration testing
- Get to know the environment(IDE, Git, Cmd)
- Introduction and understanding test-driven development

eit Digital

Agenda and Syllabus

- Throughout the 12 – 13 weeks you will learn about software quality and testing where
  - Test on Wednesday
  - Smallish tests today
  - You will need to write tests ☺
  - You'll get homework projects for(topic : topics)
  - Final exam during exam period
  - You will have fun!

eit Digital

Unit test

- Def: A unit test is a piece of a code (usually a method) that invokes another piece of code and checks the correctness of some assumption afterward. If the assumptions turn out to be wrong, the unit test has failed. A unit is a method or function.
- Def: SUT stand for system under test, and some people like to use CUT (class under test or code under test). When you test something, you refer to the thing you're testing as the SUT.

Unit test cont.

- Def: A unit of work is the sum of actions that take place between the invocation of a public method in the system and a single noticeable end result by a test of that system. A noticeable end result can be observed without looking at the internal state of the system and only through its public APIs and behavior. An end result is any of the following:
- The invoked public method returns a value
- There is a noticeable change in the state of behavior, that can be determined by interrogating the internal state
- There is a callout to a 3rd – party system where over the test has no control

Unit test cont.

- Updated def.: A unit test is a piece of code that invokes a unit of work and checks one specific end result of that unit of work, whereas it is fully isolated. If the assumptions on the end result turn out to be wrong, the unit test has failed. A unit test's scope can span as little as a method or as much as multiple classes.
- The primary goal of unit testing is to
  - Take the smallest piece of testable software in the application
  - Isolate it from the remainder of the code and
  - Determine whether it behaves exactly as you expected

Unit test cont.

- Practically, a unit test is a piece of code and
  - Invokes another piece of code
  - Checks the correctness of some assumption afterward.
  - If the assumption turn out to be wrong, the unit test has failed. A unit is a method or function.
- THE IMPORTANCE OF A GOOD UNIT TEST

Properties of a good unit test

- A unit test should have the following props.
  - It should be automated and maintainable
  - It should be easy to implement
  - It should be relevant tomorrow
  - Anyone should be able to run it at the push of a button
  - It should run quickly
  - It should be consistent in its results
  - It should be fully isolated
  - When it fails, it should be easy to detect what was expected and determine how to pinpoint the problem

Integration tests

- Def.: Integration testing is testing a unit of work without having full control over all of it and using one or more of its real dependencies, such as time, network, database, threads, random number generators, and so on.
- The complete Guide to Software testing by Bill Hetzel (Wiley, 1993), integration testing is “an orderly progression of testing in which software and/or hardware elements are combined and tested until the entire system has been integrated.”

Unite test cont.

- Def.:(final) A unit test is an automated piece of code that invokes the unit of work being tested, and then checks some assumptions about a single end result of that unit. A unit test is almost always written using a unit testing framework. It can be written easily and runs quickly. It's trustworthy, readable, and maintainable. It's consistent in it's results as long as production code hasn't changed.
- A **Simple** unit testing example

Test – driven development

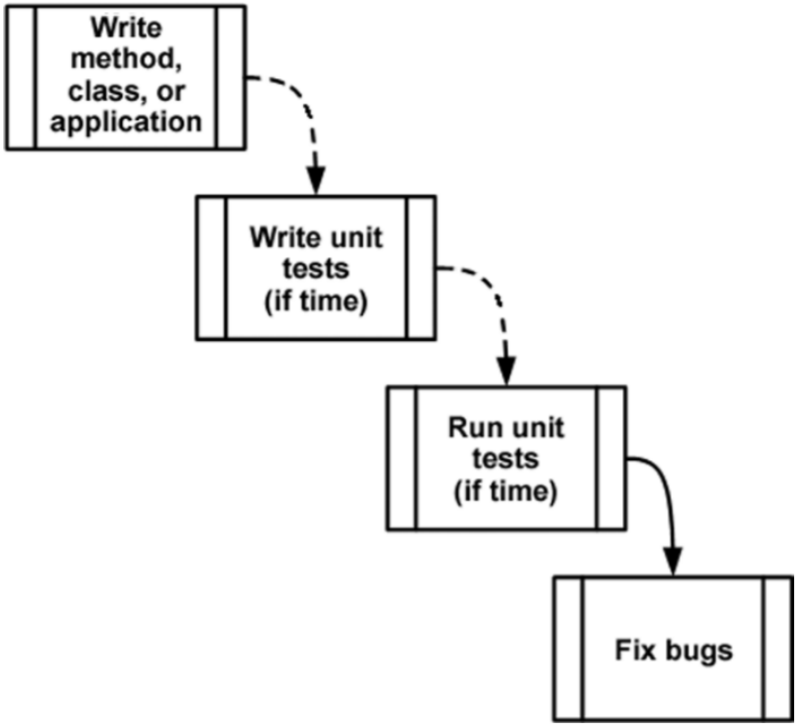

Figure 1.3 The traditional way of writing unit tests. The broken lines represent actions people treat as optional.

Test – driven development cont.

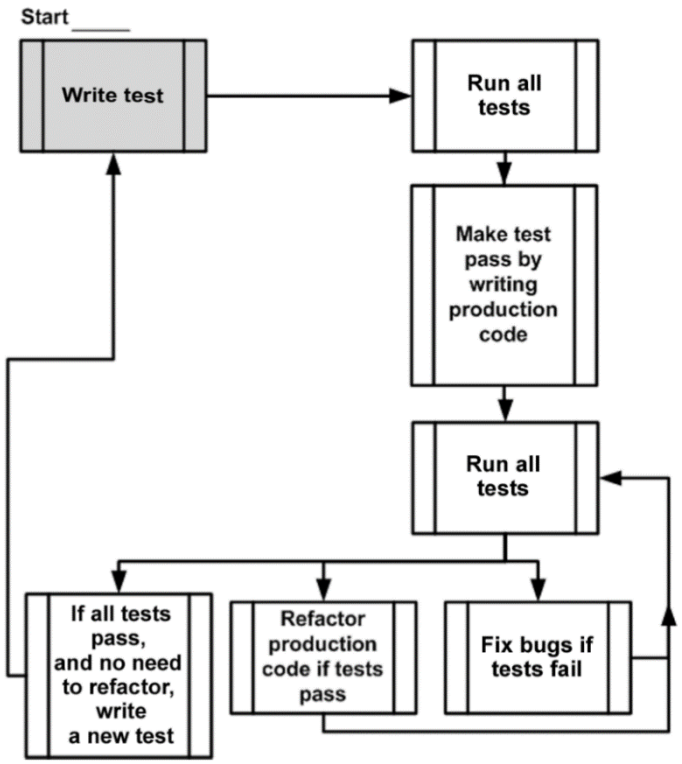

Figure 1.4 Test-driven development—a bird's-eye view. Notice the spiral nature of the process: write test, write code, refactor, write next test. It shows the incremental nature of TDD: small steps lead to a quality end result.

Technique of TDD in a nutshell

- Write failing test to prove code or functionality is missing from the end product.
- Make the test pass by writing production code that meets the expectations of your test.
- Refactor your code.

And don’t forget to make them fast

eit Digital

eitdigital.eu

http://www.doctoralschool.eitdigital.eu/doctoral-training-centres/dtc-budapest/