Software Testing, Unit testing and TDD

Software Engineering

By: Ahmed Ezzat Afifi

Contents

Software Engineering

- Software Engineering is the application of a systematic, disciplined approach to development, operation and maintenance of software.
- Software testing is an essential phase of every Software Development Life Cycle model.

Software Quality Control

- SQC is the set of procedures used by organizations to ensure that a software product will meet its quality goals at the vest value to the customers .
- Software quality control refers to specified functional requirements as well as non functional requirements.

What is Software Testing

- A software quality control activity aimed at evaluating a software item against the given system requirements.
- This include but not limited to the process of executing a program or application with the intent of finding software bugs.
- Testing is a SQC activity

Verification and Validation?

- Verification: Confirming that the software meets its specification.

- Validation: Confirming that it meets the user's requirements.

- The two major V&V activities are reviews and testing

Why Testing matters

- The software must work as intended
- Trivial bugs could cause millions of dollars
- Imperial to metric system caused NASA to lose the Mars Climate Orbiter
- List of historical software bugs with extreme consequences

What is the software “bug”?

- Things software does while it is not supposed to, OR something the software does not while it is supposed to do.

- Bugs are inevitable

What should testers do?

- Find Bugs
- Find them **early**
- Make sure they have been fixed

What testers should not do?

- Fixing Bugs
- Test with good faith!
- Test with no requirements
- Not to report bugs they found
- Mock or condescend developers for their bugs
- Again, remember bugs are **inevitable**!

Testing Life cycle

- Test planning
- Test Analysis
- Test Cases preparation
- Test Execution
- Test Reporting and monitoring
- Bugs reporting
- Release reports
- User acceptance test execution

Things to test/ Test Types

- **Black box testing** – Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.
- **White box testing** – This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.
- **Incremental integration testing** – Bottom up approach for testing i.e. continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately. done by programmers or by testers.
- **Integration testing** – Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.
- **Load testing** – Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

And…

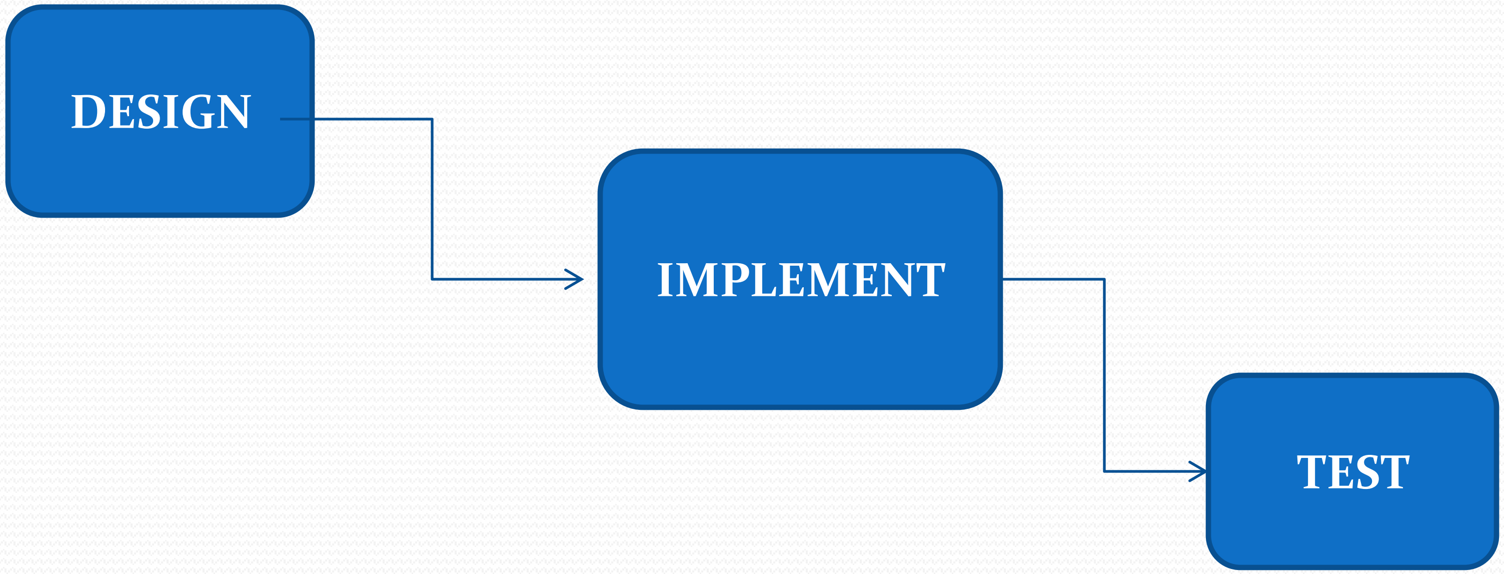
- **Unit Testing**: Testing of individual software components or modules. **Typically done by the programmer and not by testers**, as it requires detailed knowledge of the internal program design and code. may require developing test driver modules or test harnesses. I got you ☺

Test-driven development

- Extreme Programming concept, but more recently has created more general interest in its own right.
- Ensures the basic software quality

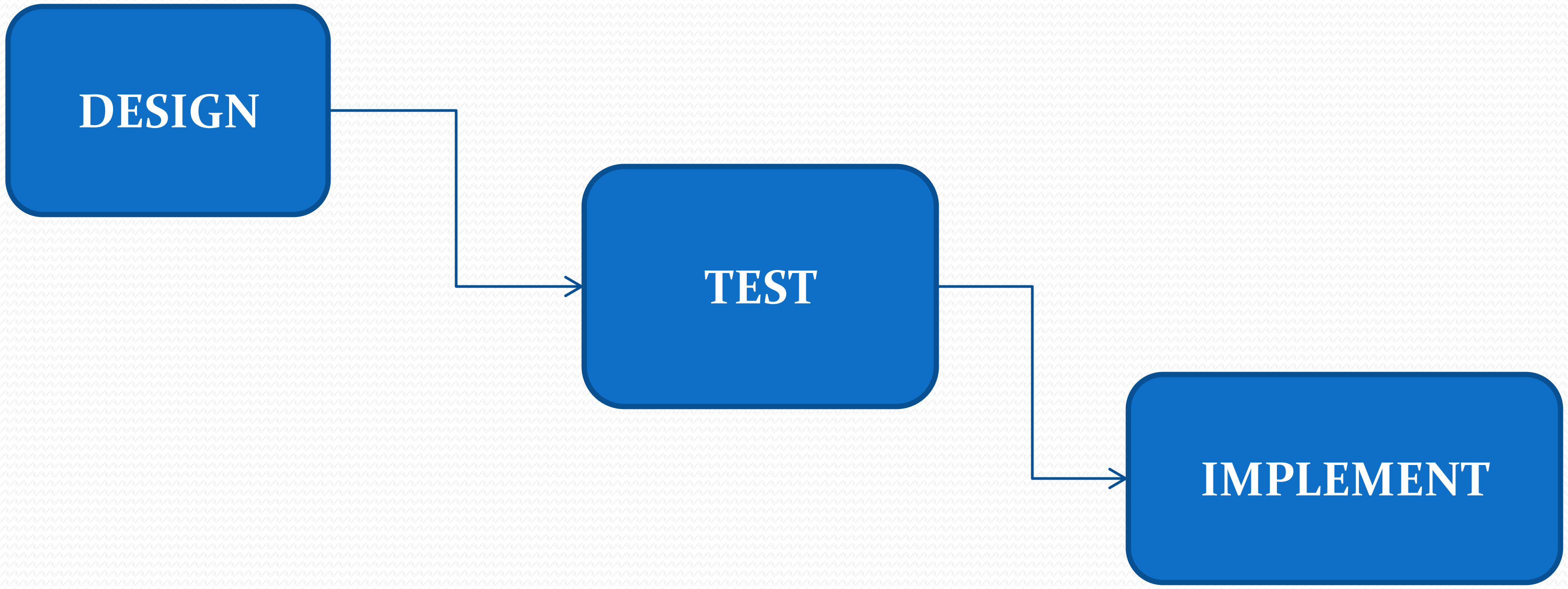
Test-Driven Development

- Usual approach



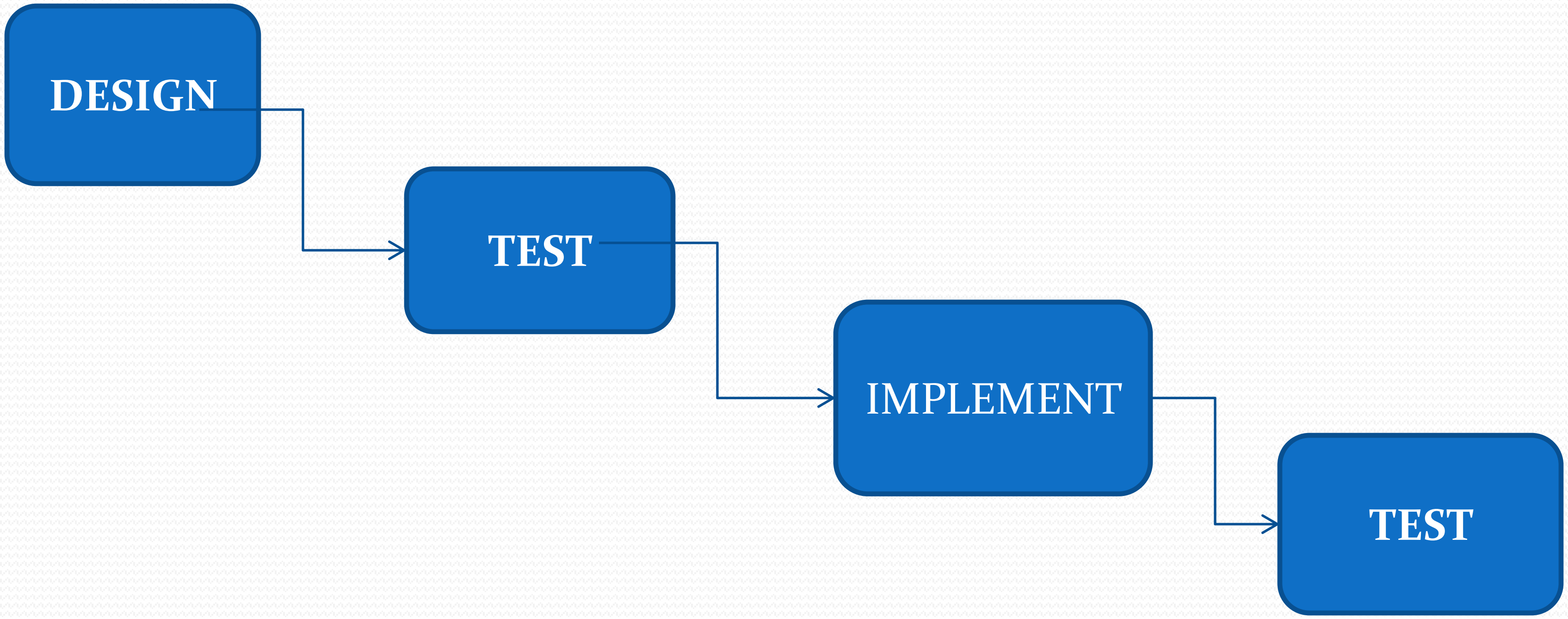
Test-Driven Development

- TDD



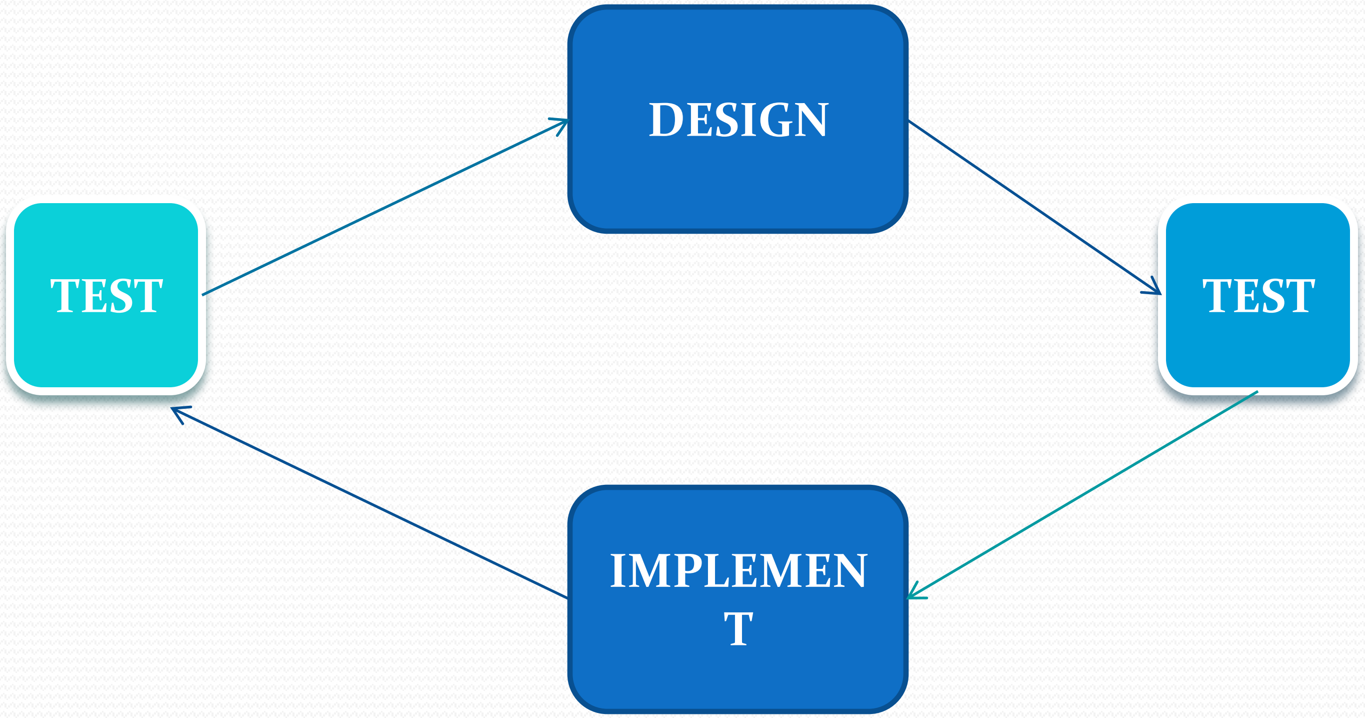
Test-Driven Development

- Well, Actually…



Test-Driven Development

- To be more honest, its an iterative process!



How to do it?

- Design: Figure out what you want to do
- Test, write test to express the design
- This test should **FAIL**
- Implement: Write the code
- Test again: This should **PASS**

Change and edit code as much as you wish!

- What if you didn’t like functions implementation?
- Does changing X will Break Y?
- Change X as you wish, if it really breaks Y, test cases will let you know!

Real example not so real

- → To code :-)

Unit Testing frameworks

- Every programming language has its own unit testing frame work, Even databases
- Most of them inherit the jUnit philosophy and architecture , known as xUnit
- List of All unit testing frameworks

xUnit Architecture

- **Test Runner**: An executable program runs the implemented test and report the results
- **Test Case**: The most element class, all xUnit inherits from here
- **Test Fixtures/Context**: set of preconditions or state to run the test, a developer sets up the good known state before running the test, and return to the original state after the test
- **Test suites**: A test suite is a set of tests that all share the same fixture. The order of the tests shouldn't matter.

Really cool stuff

- There is a unit testing framework for the Ruby programming language called Cucumber
- It enables you to write your test cases in plain English
- Not just English, any other language…. Even Arabic!

- Technically Cucumber is a DSL

Cucumber

```
# language: en
Feature: Addition
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers


Scenario Outline: Add two numbers
Given I have entered <input_1> into the calculator
And I have entered <input_2> into the calculator
When I press <button>
Then the result should be <output> on the screen


Examples:
| input_1 | input_2 | button | output |
| 20 | 30 | add | 50 |
| 2 | 5 | add | 7 |
| 0 | 40 | add | 40 |
```

Arabic Cucumber

خاصية: الجمع
من اجل تجنب الأخطاء السخيفة
كشخص غبي في الرياضيات
اريد معرفة ناتج جمع عددين

سيناريو مخطط: جمع عددين
بفرض كتابة <input_1> في الآلة الحاسبة
و كتابة <input_2> في الآلة الحاسبة
متى يتم الضغط على <button>
اذاً يظهر <output> على الشاشة

امثلة:

input_1	input_2	button	output
30	20	جمع	50
2	5	جمع	7
0	40	جمع	40

Questions?

- Thank you ☺