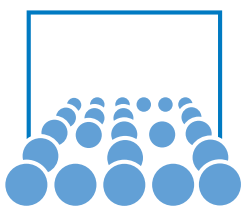
Technische Universität München

TDD: Test Driven Development

Tutorial for Advanced Programming

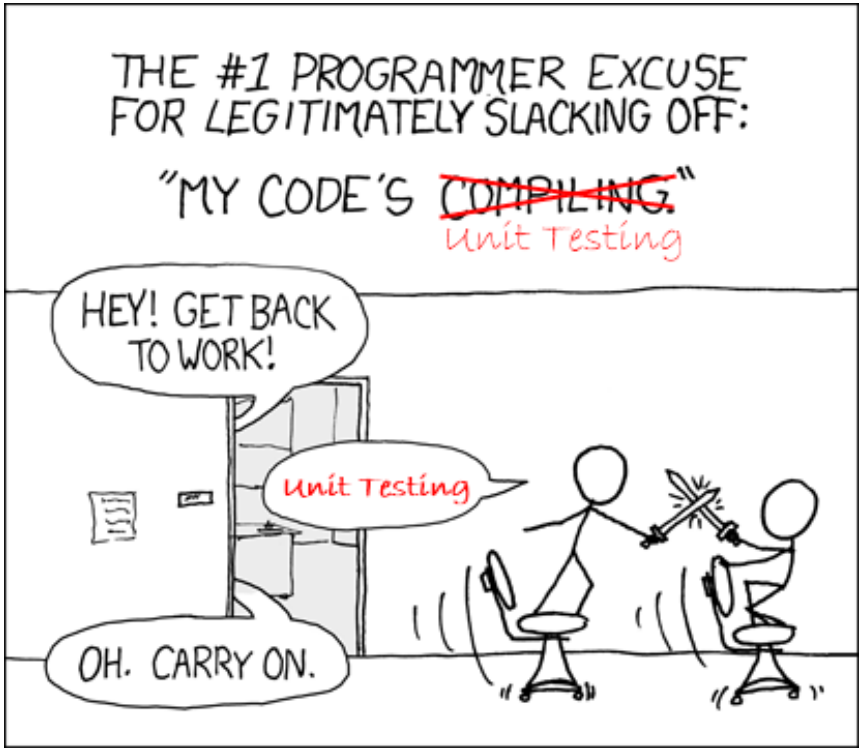
Friedrich Menhorn

November 17, 2015



Contents

Why do we need testing?



Contents

Assertions

```
#include <iostream>
double divideXByY(double x, double y){
    if(y==0){
        std::cout << "divideXByY: Division by Zero! Exit!" << std::endl;
        exit(-1);
    }
    return x/y;
}
```

Assertions

```
//#define NDEBUG //uncomment to disable
#include <assert.h>
double divideXByY(double x, double y){
    assert(y!=0); // syntax: assert(expression)
    return x/y;
}
```

Assertions

```
//#define NDEBUG //uncomment to disable
#include <assert.h>
double divideXByY(double x, double y){
    assert(y!=0); // syntax: assert(expression)
    return x/y;
}
```

- Easy and simple way to test during runtime
- Checks if `expression` is true
- On fail, exits program and returns meaningful error message
- Smallest unit to test code

Introduction to Unit Tests

> "If you're good at the debugger it means you spent a lot of time debugging. I don't want you to be good at the debugger." – *Robert C. Martin, Clean Code: A Handbook of Agile Software Craftmanship*

Introduction to Unit Tests

> "If you're good at the debugger it means you spent a lot of time debugging. I don't want you to be good at the debugger." – *Robert C. Martin, Clean Code: A Handbook of Agile Software Craftmanship*

- Code which tests single units (methods) of the code
- In OOP usually one unit test class per class/interface
- Using assertions in its core
- Executed independently of main program (overnight testing)

Unit tests in a nutshell

Unit tests in a nutshell

Unit tests in a nutshell

Unit tests in a nutshell



Unit tests in a nutshell

Unit tests in a nutshell

Runner

Unit tests in a nutshell



Runner

- ”Main” class of CppUnit
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed

Unit tests in a nutshell



Runner

- ”Main” class of CppUnit
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed

```
TestRunner tuvguy;
tuvguy.addTest(test);
tuvguy.run();
```

Unit tests in a nutshell



Runner

- ”Main” class of CppUnit
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed

```
TestRunner tuvguy;
tuvguy.addTest(test);
tuvguy.run();
```

Test suite

Unit tests in a nutshell



Runner

- ”Main” class of CppUnit
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed

```
TestRunner tuvguy;
tuvguy.addTest(test);
tuvguy.run();
```

Test suite

- Container for tests
- If run is called, runs its collection of tests

Unit tests in a nutshell



Runner

- ”Main” class of CppUnit
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed

```
TestRunner tuvguy;
tuvguy.addTest(test);
tuvguy.run();
```

Test suite

- Container for tests
- If run is called, runs its collection of tests

```
TestSuite
*carSuite = new
TestSuite("carSuite");
carSuite->
addTest(testSteering);
```

Unit tests in a nutshell



Runner

- ”Main” class of CppUnit
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed

```
TestRunner tuvguy;
tuvguy.addTest(test);
tuvguy.run();
```

Test suite

- Container for tests
- If run is called, runs its collection of tests

```
TestSuite
*carSuite = new
TestSuite("carSuite");
carSuite->
addTest(testSteering);
```

Test class

Unit tests in a nutshell



Runner

- ”Main” class of CppUnit
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed

```
TestRunner tuvguy;
tuvguy.addTest(test);
tuvguy.run();
```

Test suite

- Container for tests
- If run is called, runs its collection of tests

```
TestSuite
*carSuite = new
TestSuite("carSuite");
carSuite->
addTest(testSteering);
```

Test class

- Contains the actual tests
- `setUp()`
- `tearDown()`
- Uses assertions as core

Unit tests in a nutshell



Runner

- ”Main” class of CppUnit
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed

```
TestRunner tuvguy;
tuvguy.addTest(test);
tuvguy.run();
```

Test suite

- Container for tests
- If run is called, runs its collection of tests

```
TestSuite
*carSuite = new
TestSuite("carSuite");
carSuite->
addTest(testSteering);
```

Test class

- Contains the actual tests
- `setUp()`
- `tearDown()`
- Uses assertions as core

```
void
testSteering();
```

”But why should I use unit tests?”

Advantages of unit tests

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

⊕ Code less error prone

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

- ⊕ Code less error prone
- ⊕ Makes you think about your design

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

- ⊕ Code less error prone
- ⊕ Makes you think about your design
- ⊕ Faster debugging

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

- ⊕ Code less error prone
- ⊕ Makes you think about your design
- ⊕ Faster debugging
- ⊕ ”Fear of change” decreases

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

- ⊕ Code less error prone
- ⊕ Makes you think about your design
- ⊕ Faster debugging
- ⊕ ”Fear of change” decreases
- → Fault detection

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

- ⊕ Code less error prone
- ⊕ Makes you think about your design
- ⊕ Faster debugging
- ⊕ ”Fear of change” decreases
- → Fault detection
- → Fault avoidance

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

- ⊕ Code less error prone
- ⊕ Makes you think about your design
- ⊕ Faster debugging
- ⊕ ”Fear of change” decreases
- → Fault detection
- → Fault avoidance

} Quality Control

”But why should I use unit tests?”

Advantages of unit tests

> “Why do most developers fear to make continuous changes to their code? They are afraid they’ll break it! Why are they afraid they’ll break it? Because they don’t have tests.” – *Robert C. Martin, The Clean Coder: A Code of Conduct for Professional Programmers*

- ⊕ Code less error prone
- ⊕ Makes you think about your design
- ⊕ Faster debugging
- ⊕ ”Fear of change” decreases
- → Fault detection
- → Fault avoidance

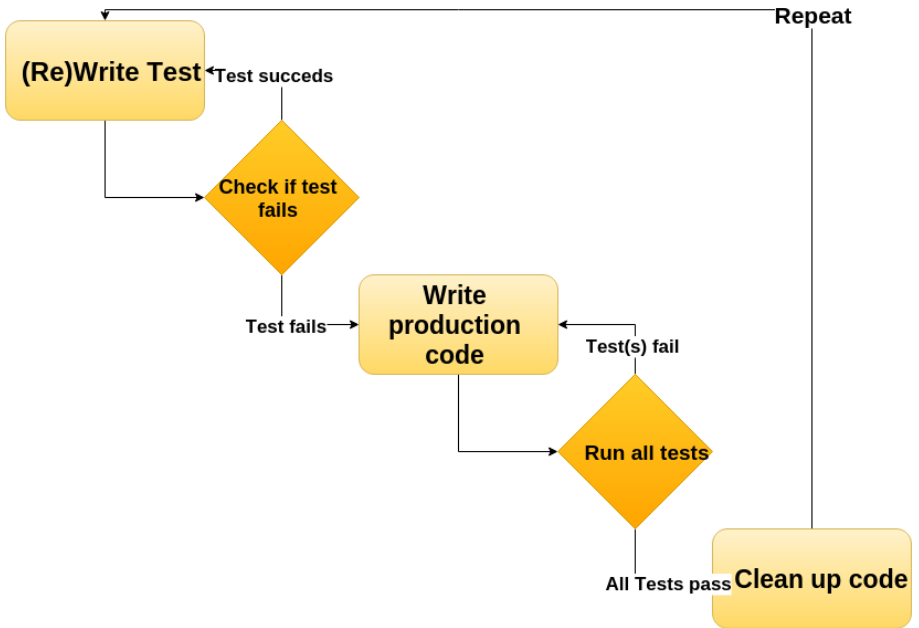
} Quality Control

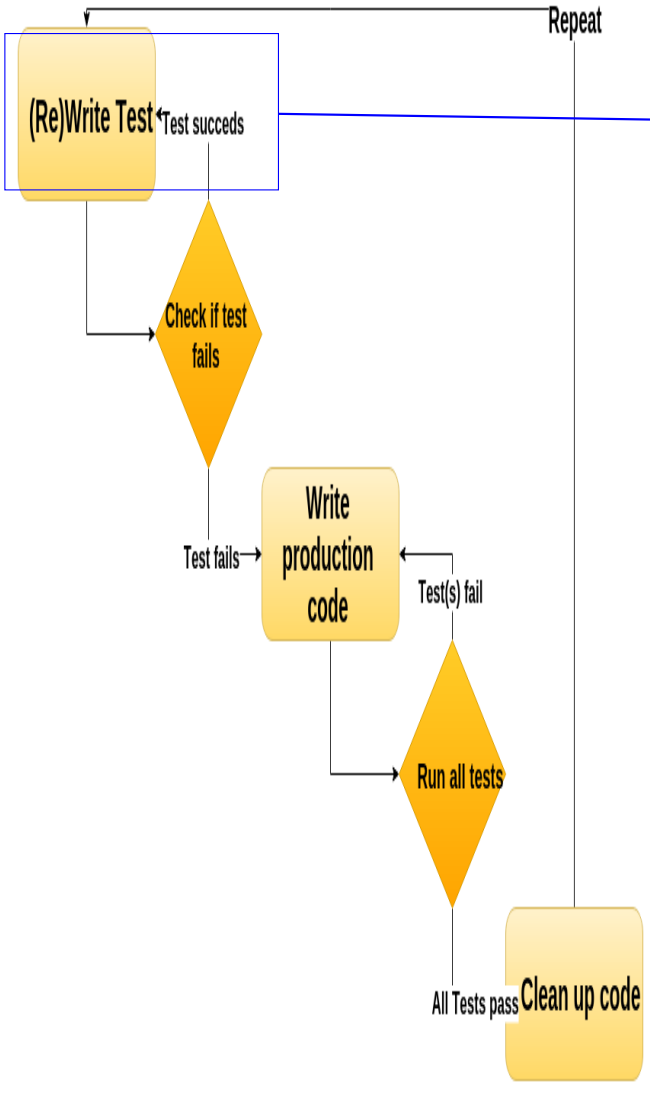
- ⇒ Cleaner development

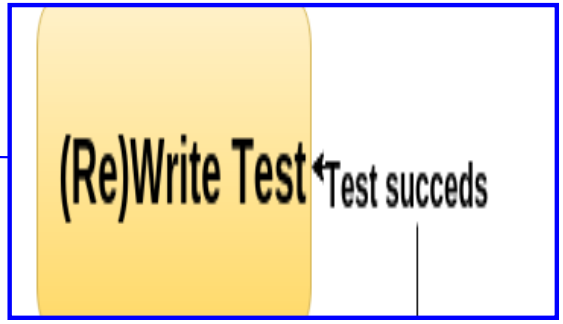
Contents

The three laws of TDD [Clean Code: Robert C. Martin]

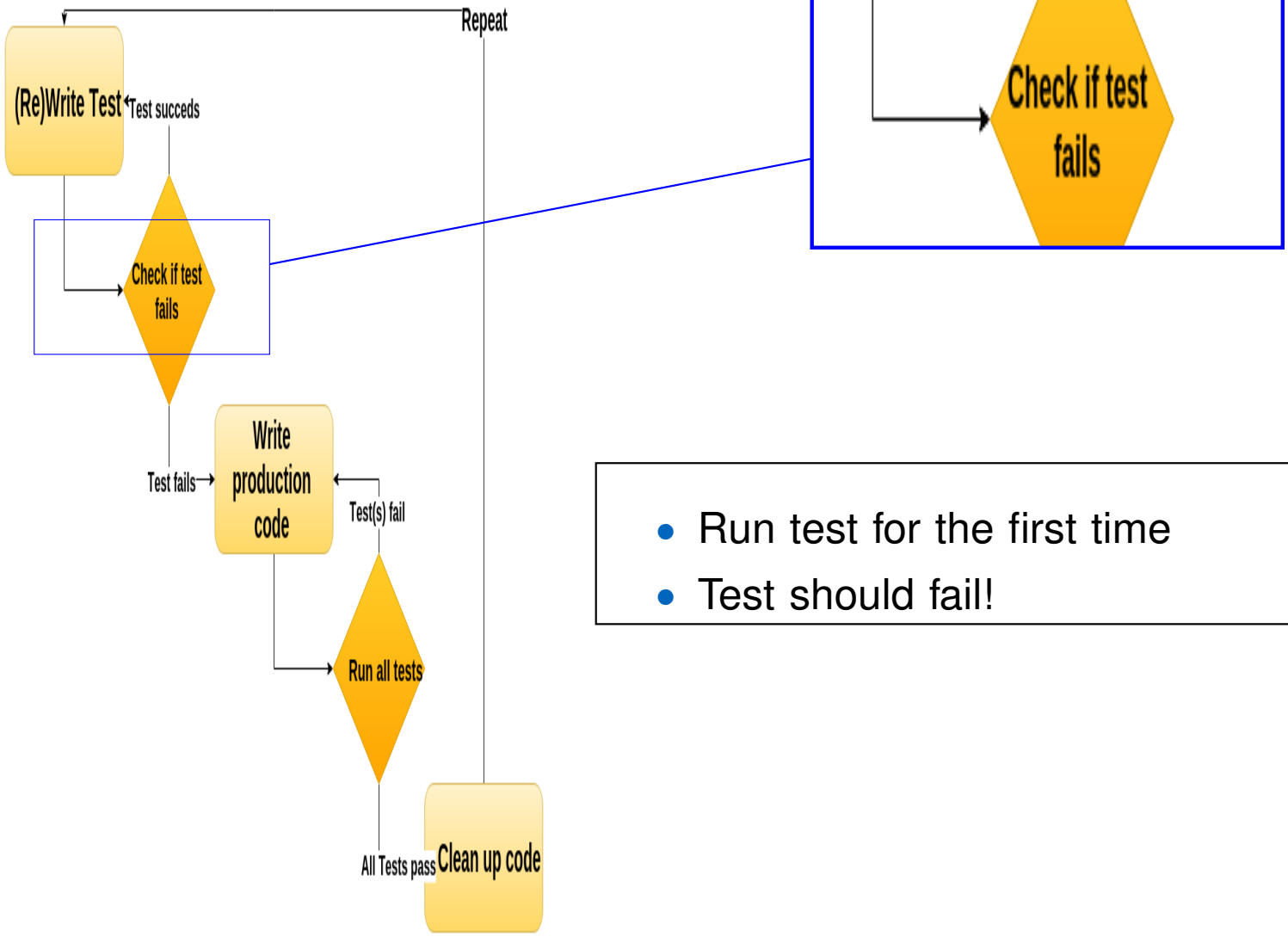
1. You may not write production code until you have written a failing unit test.
2. You may not write more of a unit test than is sufficient to fail, and not compiling is failing.
3. You may not write more production code than is sufficient to pass the current failing test.



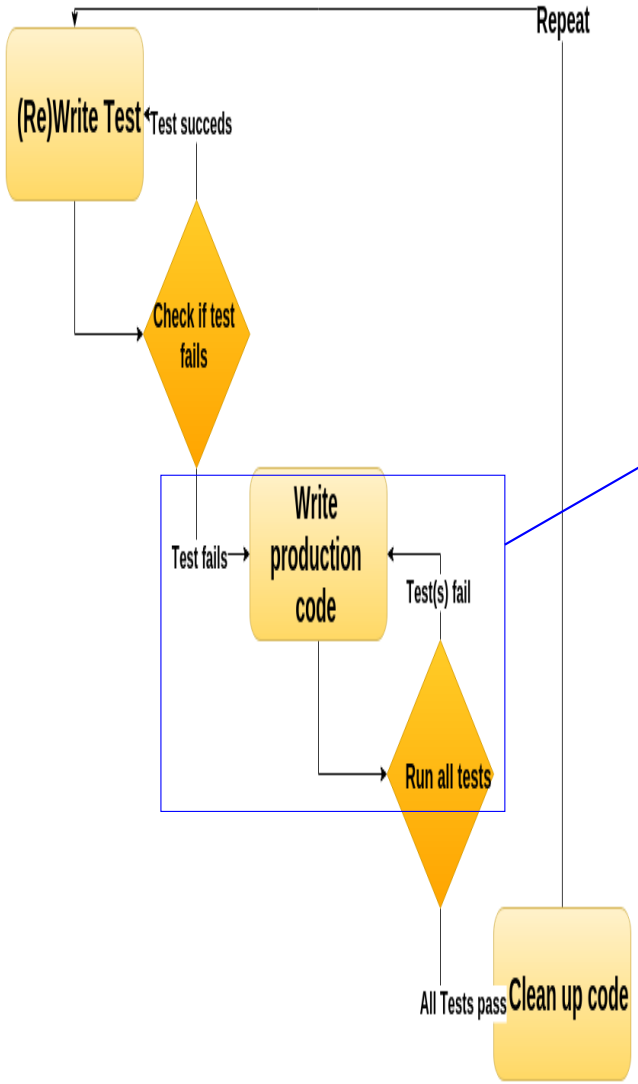


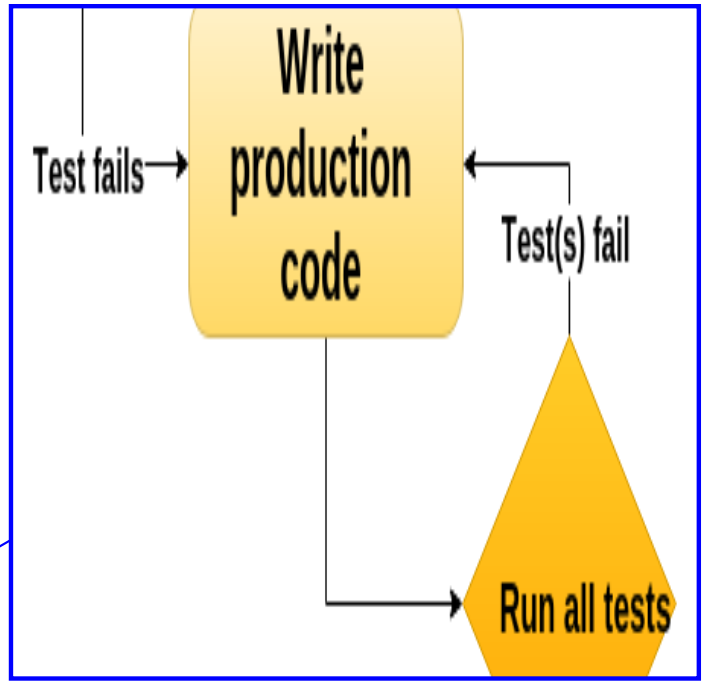


- Before writing any code, first write an automated test for your code.
- Take into account all possible inputs, errors, and outputs.

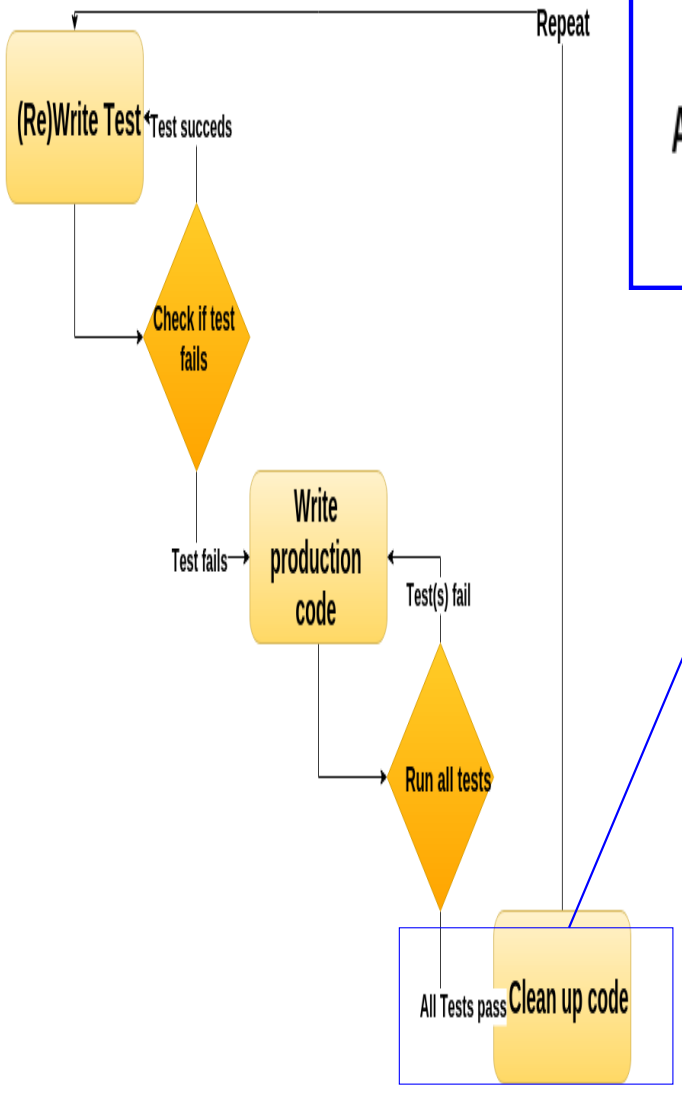


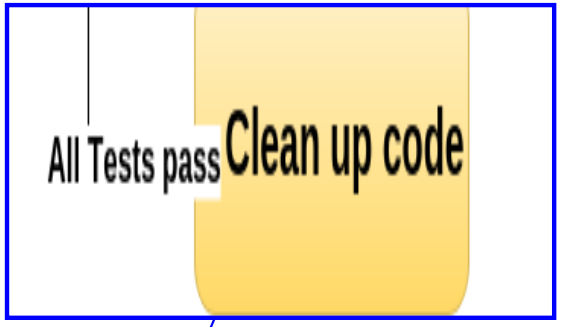
- Run test for the first time
- Test should fail!





- Begin programming...
- As long as the code fails new test, it is still not ready
- Fix code until all assertions are passed



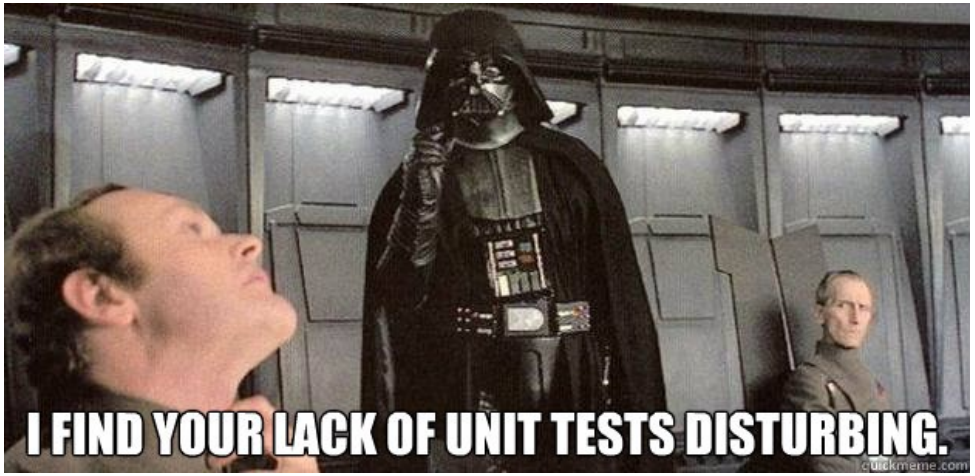


- Once all tests are passed, start clean up code
- If tests are passed $\rightarrow$ code is working
- Repeat

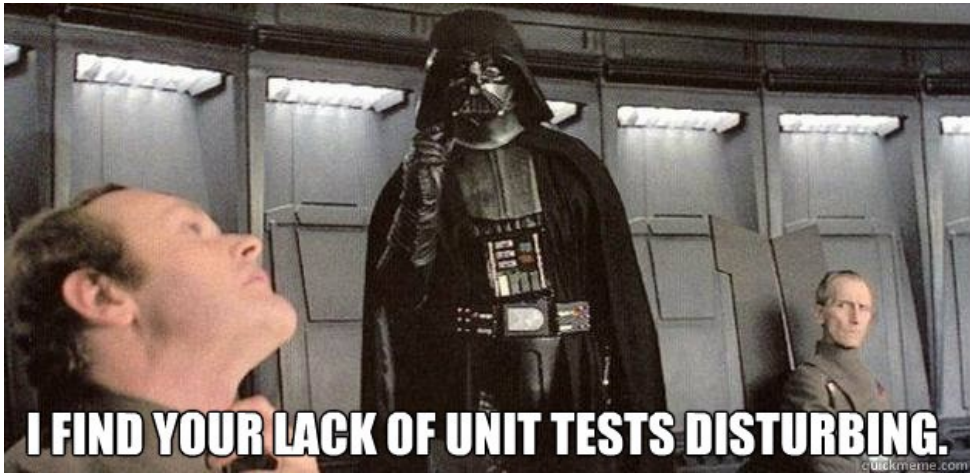
Contents

Let’s ask the internet about unit testing

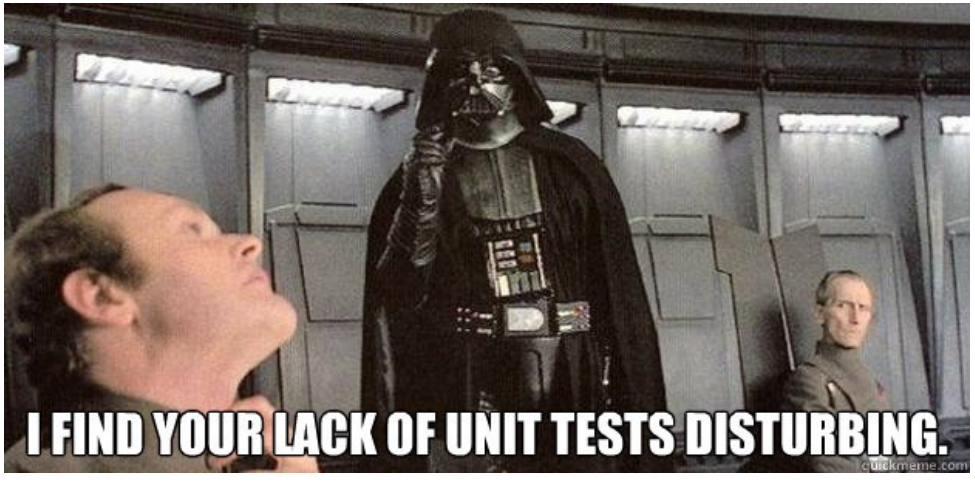
Let’s ask the internet about unit testing

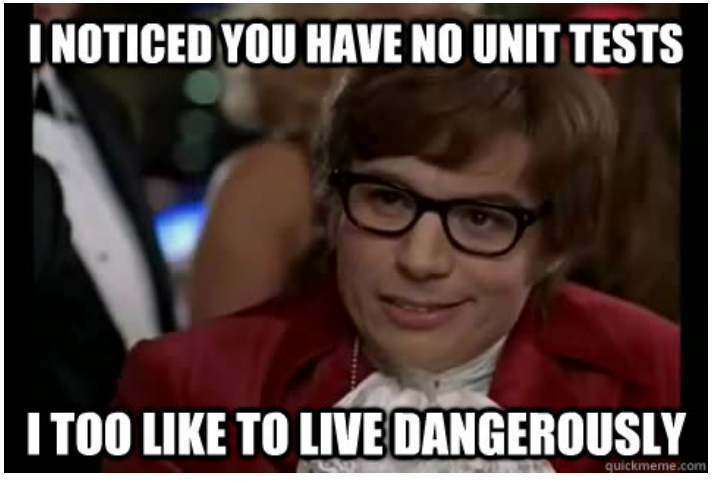


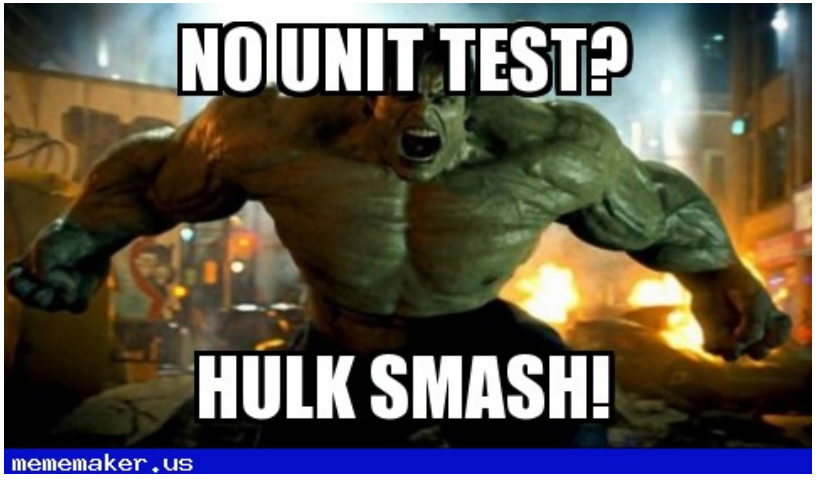
Let’s ask the internet about unit testing



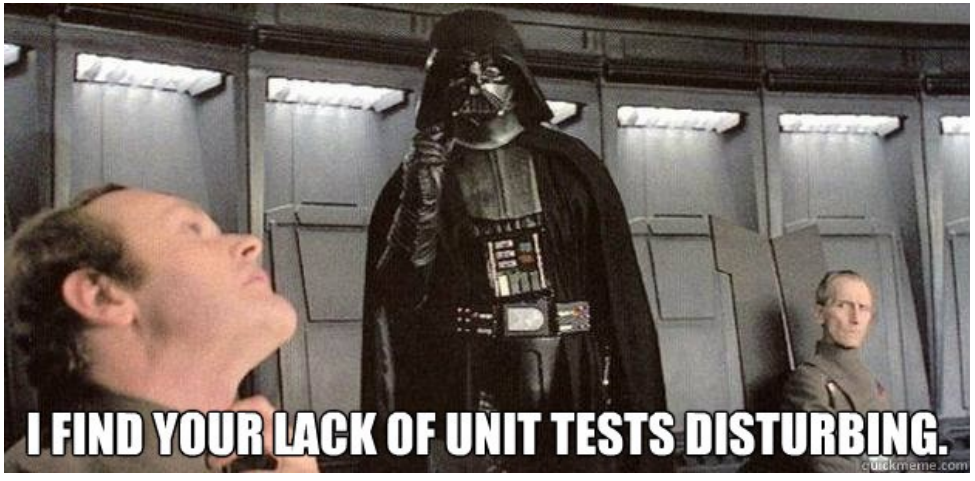
Let’s ask the internet about unit testing

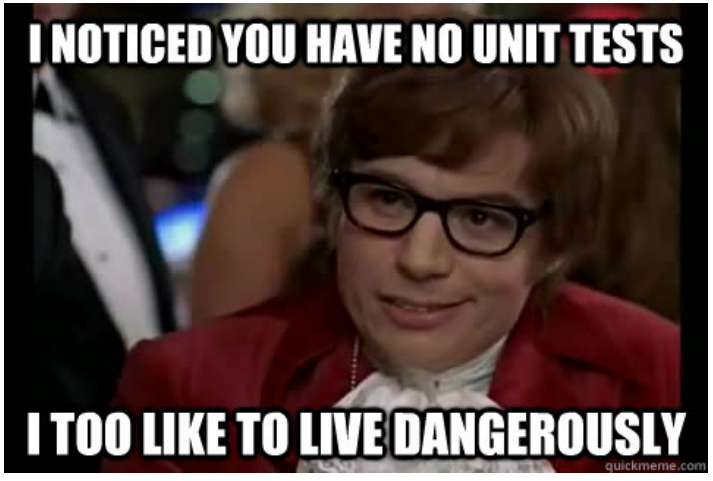


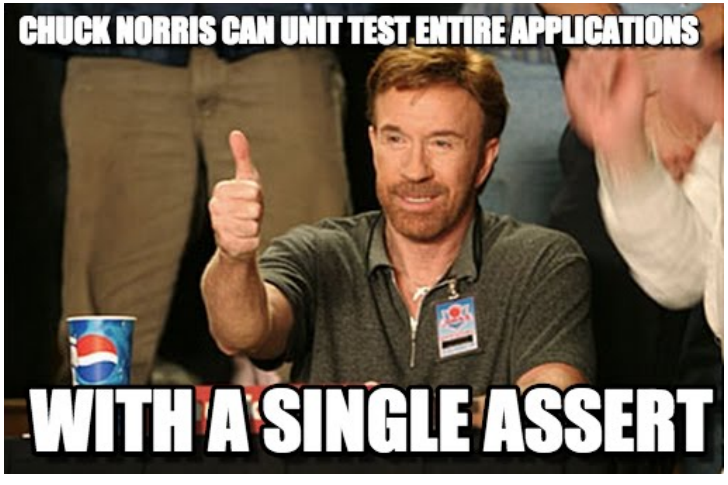


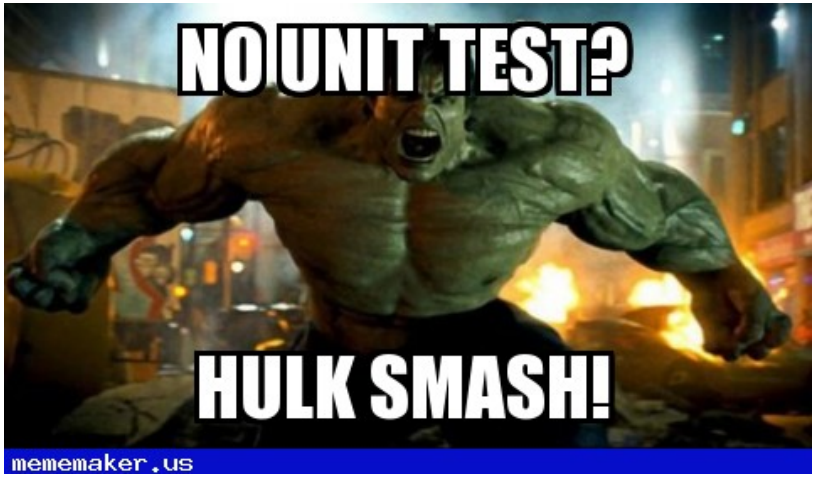


Let’s ask the internet about unit testing

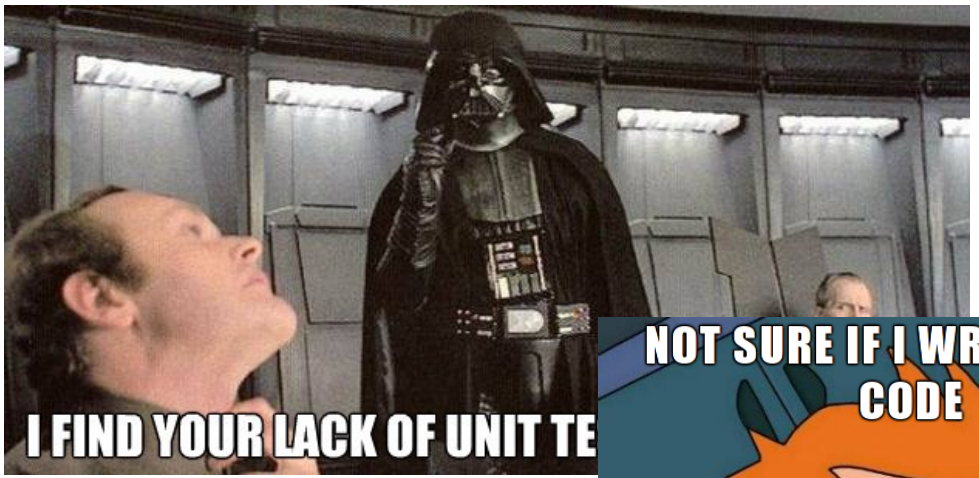


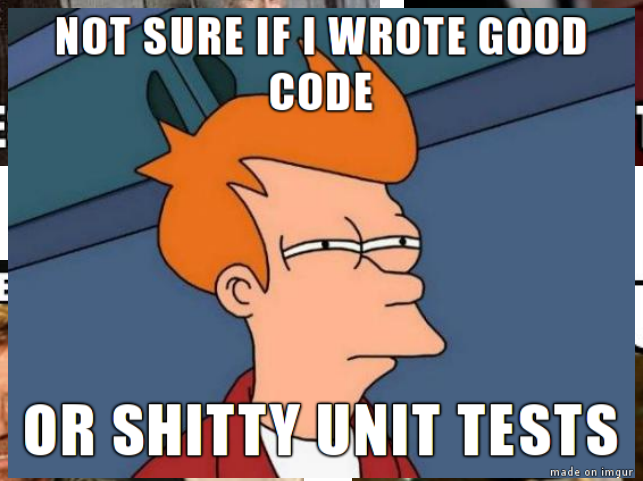


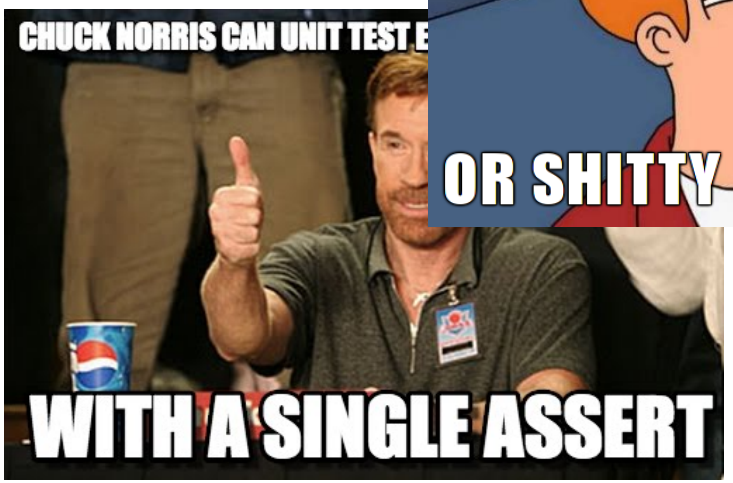


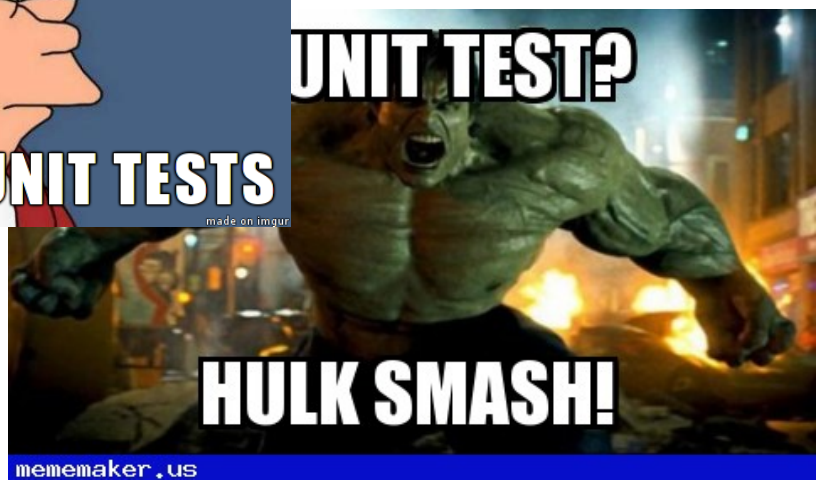


Let's ask the internet about unit testing









Contents

References for unit tests in C++

- CppUnit: `http://sourceforge.net/projects/cppunit`
- Eclipse CDT (usually not installed by default): `http://stackoverflow.com/questions/16741400/eclipse-cdtplugin-for-running-tests-and-browsing-report`
- Igloo C++ testing framework: `http://igloo-testing.org`
- Google C++ testing framework: `http://code.google.com/p/googletest`
- CUTE: `http://cute-test.com`

CppUnit

- CppUnit: `http://sourceforge.net/projects/cppunit`
- Port from JUnit
- Installation:
  - `sudo apt-get install libcppunit-doc libcppunit-dev`
  - Include linker flag: `-lcppunit`

Structure of CppUnit

Runner class

- ”Main” class of CppUnit
- Sets up the basic test environment
- Manages the life cycle of the added tests
- Runs the test
- Prints out a trace as the tests are executed followed by a summary at the end (optional)

```
Test* test = new CppUnit::TestCaller<your_test>(
    "testA",
    &your_test::testA);
CppUnit::TextUi::TestRunner runner;
runner.addTest(test);
runner.run();
```

Structure of CppUnit

Test suite

- Container for tests
- If run is called, runs its collection of tests

```
CppUnit::TestSuite *test_suite =
new CppUnit::TestSuite( "Name_Of_TestSuite" );
test_suite->addTest( test );
```

Structure of CppUnit

Test class

- Contains the actual tests
- `setUp()`:
  - Called before each test
  - Sets up test environment
- `tearDown()`:
  - Called after each test
  - Destroys test environment
- Uses assertions as core

```
1 class your_test: public CppUnit::TestFixture {
  void setUp();
3 void tearDown();
  void testA();
5 void testB();
  }
```

Structure of CppUnit

Add test

```
test_suite->addTest(
new CppUnit::TestCaller<your_test>(
    "testA",
    &your_test::testA)   );
```

Contents

- Slide5: `http://www.memegenerator.net`
- Tüvguy, Slide7: `http://kfz-meisterbetrieb-bauer.de/wp-content/uploads/2013/04/tuev1.jpg`
- Car, Slide7: `http://s3-ap-southeast-1.amazonaws.com/first.code.academy.hk/uploads/image/image/4086/1.png`
- Steering Wheel, Slide7: `http://i.telegraph.co.uk/multimedia/archive/02625/Steering-wheel_2625587b.jpg`
- Slide8: `http://www.memegenerator.net`
- TDD, Slide11: `https://en.wikipedia.org/wiki/Test-driven_development`
- TDD, Slide14: `http://www.memegenerator.net`