Remark: TODO: Change \iftrue in commands.tex \def\rem to \iffalse in final copy

COMP2123 self-learning report
Unit testing

December 17, 2018

Abstract

This report goes through the motivation, frameworks used and difficulties of unit testing.

Contents

1 What is unit testing? **4**

2 Motivation **4**
- 2.1 Discover bugs early 4
- 2.2 As a method of specification 4

3 Unit testing methods **4**
- 3.1 Testing for expected result 4
- 3.2 Generating test parameters 4
  - 3.2.1 Testing for edge cases 4
- 3.3 Test case selection 4

4 Unit testing techniques **5**
- 4.1 Test coverage 5
- 4.2 Test-driven development (TDD) 5
- 4.3 Behaviour-driven development (BDD) 5
- 4.4 Dependency mocking 5

5 Coupling between units **5**

1 What is unit testing?

Unit testing is the practice of testing every small unit of a large project instead of only testing the output.

2 Motivation

2.1 Discover bugs early

As the scale of a software project grows, debugging becomes more complicated. It may take a long time to discover edge case bugs in an old component, which is very difficult to debug after a long time. Unit testing allows identification of bugs as soon as possible with little impact.

2.2 As a method of specification

Unit testing can also be used as a means of project requirement specification.

3 Unit testing methods

3.1 Testing for expected result

The intuitive way is to write a test that tests each function.

```
class SimpleSpec {
public:
    void testFooBar() {
        ASSERT_EQUAL(fooBar(), "qux")
    }
}
```

The `ASSERT_EQUAL` macro function would compare the result of `fooBar()` with `"qux"` and trigger an error if they are not equal.

3.2 Generating test parameters

Remark: Generate parameters randomly, possibly by reverse calculation

3.2.1 Testing for edge cases

Remark: E.g. test for empty strings, Float.INFINITY, 0, etc.

3.3 Test case selection

As the number of variables to a feature increases, the number of test cases might increase exponentially.

4 Unit testing techniques

4.1 Test coverage

Test coverage is a criterion to assess the representativeness of the unit tests of a project by counting the number of lines executed in the test.

Remark: Talk about the integration of debuggers and how to interpret coverage

4.2 Test-driven development (TDD)

As a consequence of unit testing, development flow becomes more fluent if each development subtask is based on certain test cases.

4.3 Behaviour-driven development (BDD)

Remark: Refer to cucumber

4.4 Dependency mocking

Remark: https://enterprisecraftsmanship.com/2016/06/09/styles-of-unit-testing/ provides some insight on why mocking is bad

5 Coupling between units