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 Motivation

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 Unit testing methods

2.1 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.

2.2 Generating test parameters

2.3 Testing for edge cases

2.4 Test case selection

3 Unit testing tools

3.1 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.

3.2 Behaviour-driven development

4 Modular coupling

4.1 Dependency mocking