Verifying Behaviors

Dror Helper

@dhelper http://blog.drorhelper.com

Module Overview

Checking method calls

State testing vs. interaction testing

Explicit vs. implicit verification

GMock and other testing frameworks

```
EXPECT_CALL(myMock, SomeMethod(42)).WillOnce(Throw(meaningException))

EXPECT_CALL(myMock, SomeMethod(_)).WillRepeatedly(Return(“42”));
```

Recap: Setting Behaviors on Fake Objects

Using EXPECT_CALL macro

- Return value
- Throw exception
- Invoke custom code

Setting Expectations Using EXPECT_CALL

```
EXPECT_CALL(mock, method(matchers))
                   .With(multi argument matchers)
                   .Times(cardinality)
                   .InSequence(S1..Sn)
                   .After(expectations)
                   .WillOnce(action)
                   .WillRepeatedly(action)
                   .RetireOnSaturation();
```

Verifying Behaviors

```
EXPECT_CALL(myFake, MyFunc()).Times(3);
EXPECT_CALL(myFake, MyFunc()).Times(Exactly(3));
EXPECT_CALL(myFake, MyFunc()).Times(3).WillOnce(Return(10));


EXPECT_CALL(myFake, MyFunc()).Times(AtLeast(1));
EXPECT_CALL(myFake, MyFunc()).Times(AtMost(3));
EXPECT_CALL(myFake, MyFunc()).Times(Between(1, 3));
EXPECT_CALL(myFake, MyFunc()).Times(AnyNumber());
EXPECT_CALL(fake, Method).Times(AtLeast(1))

EXPECT_CALL(fake, Method).Times(Exactly(0)) // or Times(0)
```

Avoid Over Specification

Unless part of the business rules

Most of the time needs to verify that the method

- Was called one or more times
- Was never called

State Based Testing VS Interaction Testing

State Based Testing	Interaction Testing
“Classic”	“Mockist”
Returned value or object state	Method was/wasn’t called
Using Assertions	Using Fakes/Mocks

“Mockist tests are thus more coupled to the implementation of a method. Changing the nature of calls to collaborators usually cause a mockist test to break.”

Martin Fowler

When to Test Interactions?

Test result is not accessible

Test result is external to the system under test

Business Requirements

Naggy, Nice and Strict Mocks

```
// Show warnings for uninteresting calls
FakeRestApiClient naggy_fakeClient


// Ignore all uninteresting calls
NiceMock<FakeRestApiClient> nice_fakeClient


// All uninteresting calls become failures
StrictMock<FakeRestApiClient> strict_fakeClient
```

Verifying Calls Are Made in the Correct Order

```
Expectation init_x = EXPECT_CALL(foo, InitX());
Expectation init_y = EXPECT_CALL(foo, InitY());


EXPECT_CALL(foo, Bar()).After(init_x, init_y);
```

```
ExpectationSet all_inits;
for (int i = 0; i < element_count; i++) {
    all_inits += EXPECT_CALL(foo, InitElement(i));
}
EXPECT_CALL(foo, Bar()).After(all_inits);
```

Verifying Call Order Using Sequences

```
{
    InSequence sequence;

    EXPECT_CALL(fake, MyMethod(1));
    EXPECT_CALL(fake, MyMethod(2)).Times(2);
    EXPECT_CALL(fake, OtherMethod(_));
}
```

Verifying Partially Ordered Calls

```
{
    Sequence s1, s2;

    EXPECT_CALL(fake, MyMethod(1)).InSequence(s1, s2);
    EXPECT_CALL(fake, MyMethod(2)).InSequence(s1);
    EXPECT_CALL(fake, OtherMethod(_)).InSequence(s2);
}
```

Controlling Expectations Lifecycle

```
EXPECT_CALL(fakeClient, HttpGet(_))
                .WillOnce(Return(movieMeta));


EXPECT_CALL(fakeClient, HttpGet(_))
                .Times(1)
                .WillOnce(Return(movieList));
```

Controlling Expectations Lifecycle

```
EXPECT_CALL(fakeClient, HttpGet(_))
                .WillOnce(Return(movieMeta));


EXPECT_CALL(fakeClient, HttpGet(_))
                .Times(1)
                .WillOnce(Return(movieList));
                .RetiresOnSaturation();
```

Verifying the order of method calls leads to fragile tests that depend heavily on implementation

Using VerifyAndClear

```
TEST(MyFixture, MyTest)

{

    MockObject mock;


    EXPECT_CALL(mock, DoThat(_))
                .Times(AtLeast(1)).WillRepeatedly(Return(5));

    // Test Code


    Mock::VerifyAndClear(&mock);

}
```

Verifying and Resetting a Mock

```
// Verify and removes expectations
Mock::VerifyAndClearExpectations(&mock);

// Also removes default actions (ON_CALL)
Mock::VerifyAndClear(&mock);

// Do not verify object
Mock::AllowLeak(&mock)
```

Summary

Verifying method was called/not called

Naggy, Nice and Strict

Testing call order

Explicit verifications

GMock and other testing frameworks

How to avoid over specifications