The Sector Specialists

Mocks

Prepared by: (Sebastian Kaczmarkiewicz)

Confidential

Agenda

CLIENT LOGO
(edit Slide
Master)

- Dependencies
- Testable code
- Mock vs Stub
- Mocks frameworks
- Moq
- Summary
- References
- Q & A

Dependencies

CLIENT LOGO (edit Slide Master)

Dependency is an object in your system that your code under test interacts with, and over which you have no control.

Typical examples:

- Filesystem
- DB connection
- any external system references

Testable code

CLIENT LOGO (edit Slide Master)

```
...
Dependency dep = new Dependency();
dep.GetData();
...
```

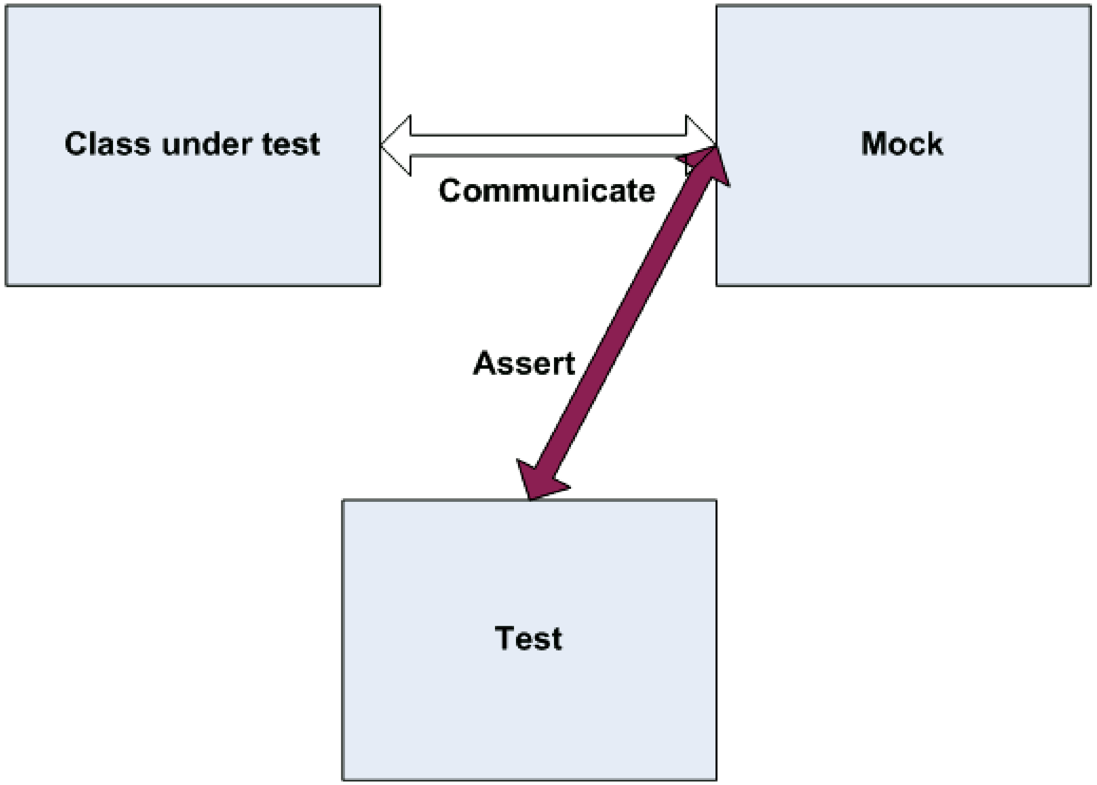
A need for layer of inderection to break dependency. (Mostly by using Interface)

- Receive an interface at the constructor level and save it in a field for later use.
- Receive an interface as a property get or set and save it in a field for later use.
- Receive an interface just before the call in the method under test using
  - a parameter to the method (parameter injection).
  - a factory class.
  - a local factory method.

Mock vs Stub
Mock

CLIENT LOGO (edit Slide Master)

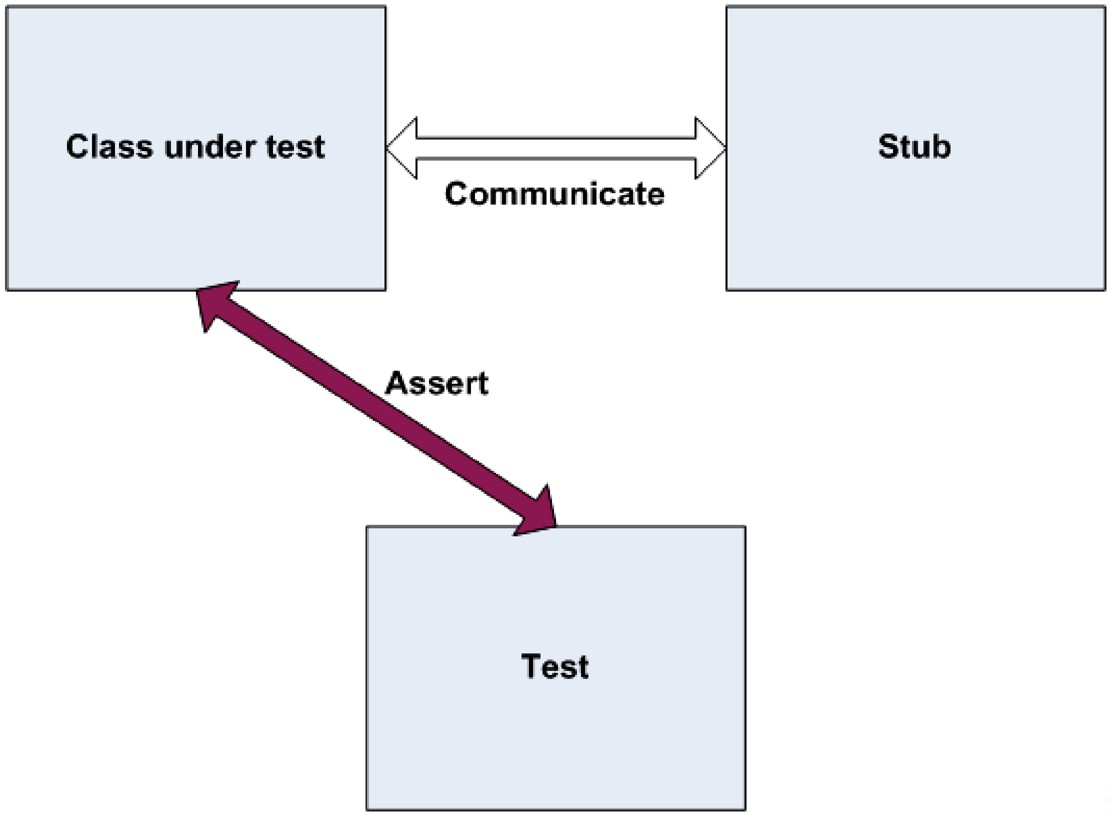
The class under test communicates with the mock object. The test uses the mock object to verify that the test passes.



Mock vs Stub
Stub

CLIENT LOGO (edit Slide Master)

The assert is performed on the class under test. The stub aids in making sure the test runs smoothly.



Mock frameworks

CLIENT LOGO
(edit Slide Master)

- Creating mocks and stubs easier
- Dynamic mocks
- Syntax: Record and replay, Arragnge-Act-Assert
- Parameter verification
- Multiple method calls verification

Frameworks:

- Microsoft Fakes – integrated with VS2012 Ultimate, no real mocks, (stub, mole, shim)
- Nmock – ported jMock, string for method name expectations
- TypeMock - compile-time mock checking, intercept the calls ant return own value
- Rihno Mocks – easy to use, free
- Moq – compile-time mock checking, free

Moq

CLIENT LOGO
(edit Slide
Master)

- Based on .NET 3.5 (utilize lambda expression)
- Free of charge
- Strong-typed
- Mock both interfaces and classes
- State-based testing and interaction testing
- Override expectations
- Intercept and raise events on mocks

Moq example

CLIENT LOGO
(edit Slide
Master)

```
var mock = new Mock<IFoo>();
mock.Setup(foo => foo.DoSomething("ping")).Returns(true);


// access invocation arguments when returning a value
mock.Setup(x => x.DoSomething(It.IsAny<string>())).Returns((string s) => s.ToLower());
// Multiple parameters overloads available


// access invocation arguments
mock.Setup(foo => foo.Execute(It.IsAny<string>())).Returns(true).Callback((string s) => calls.Add(s));


// Called at least once
mock.Verify(foo => foo.Execute("ping"), Times.AtLeastOnce());


// Raising an event on the mock
mock.Raise(m => m.FooEvent += null, new FooEventArgs(fooValue));
```

Summary

CLIE
(ed
M

- Stub and Mock
- Verification (parameter method calls)
- Creation of Mocks

References

- The Art of Unit Testing
- Moq

Example Q & A

CLIENT LOGO
(edit Slide
Master)

Q & A