NOVIUS
The e-novative company

Testing

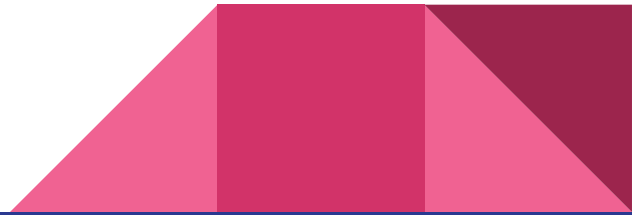
July 2017

Abraham Mesa
Full stack developer

github.com/defnx
abrahammesa.com
defn.es

Testing is the process of executing a program with the intent of finding errors. [...] This definition of testing has many implications [...] It implies that testing is a destructive process, even a sadistic process, which explain how many people find it difficult.

Glenford J. Myers



Why write tests?

1

It is your responsibility

2

The only way to make a refactor

3

Good live documentation of the behaviour of the system

4

It is a design tool

TDD

5

Show me the MONEY

6

Validates the system

7

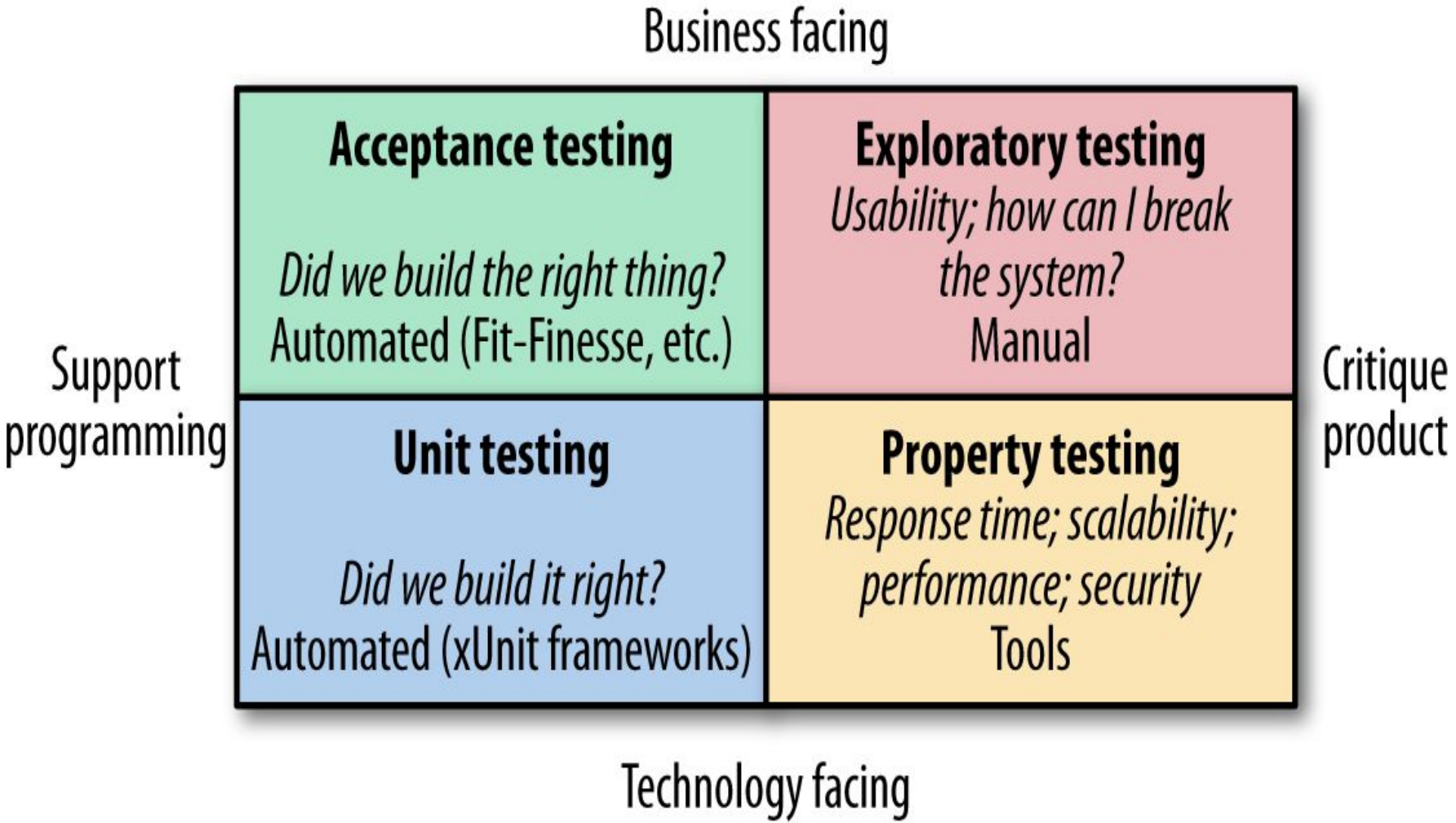
Customer acceptance

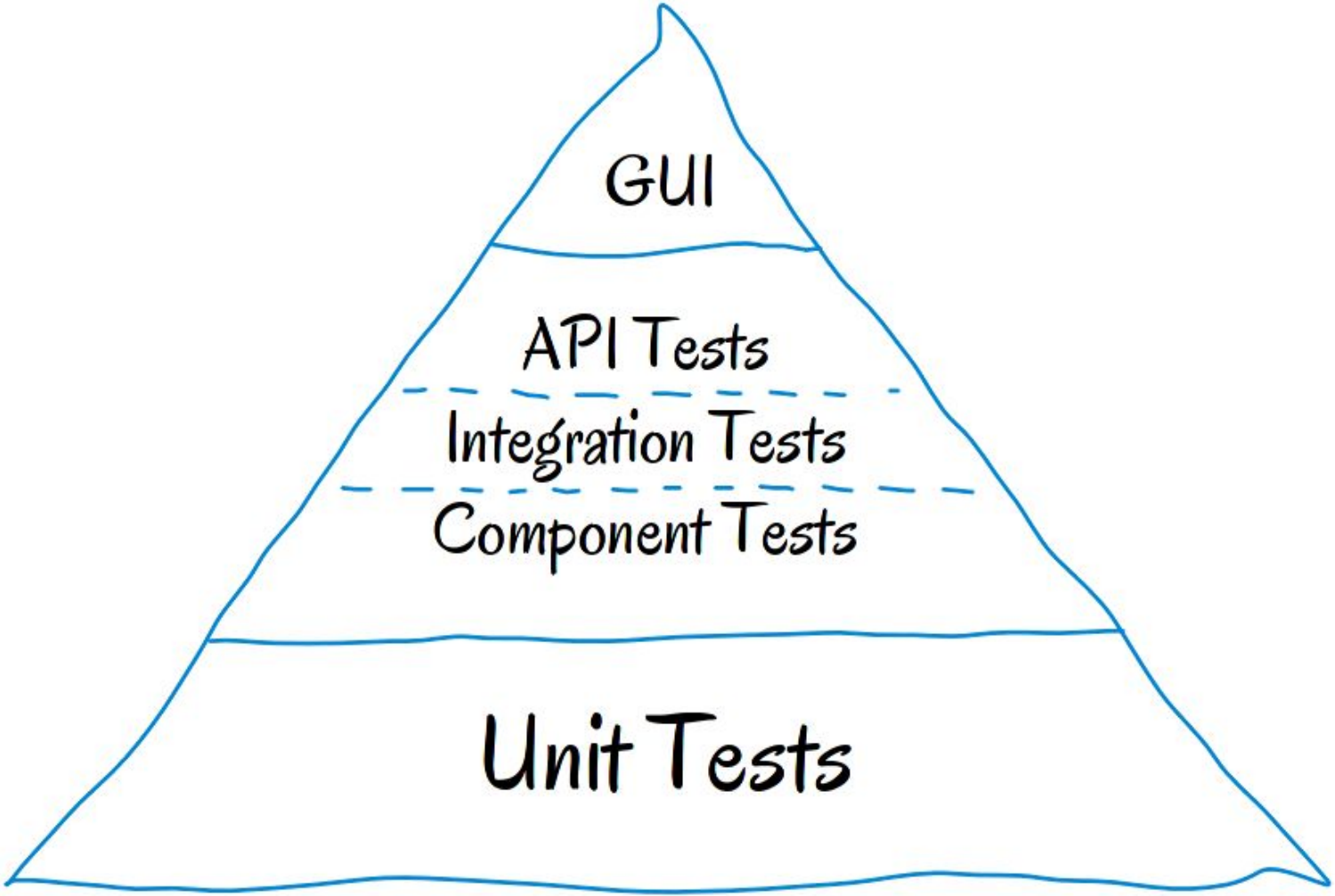
8

Flow

“Bad code affect to your customers”

Martin Fowler


Business facing
Acceptance testing
Did we build the right thing?
Automated (Fit-Finesse, etc.)
Exploratory testing
Usability; how can I break the system?
Manual
Support programming
Critique product
Unit testing
Did we build it right?
Automated (xUnit frameworks)
Property testing
Response time; scalability; performance; security
Tools
Technology facing


GUI
API Tests
Integration Tests
Component Tests
Unit Tests

Acceptance tests

From the point of view of the customer

Unit tests

24 definitions of unit test

Type of tests

State verification

Behaviour verification

- Unit tests should be isolated from its collaborators.
- Small scope.
- Unit tests are not silver bullets.
- A unit is not a class.

Fast
Isolated
Repeatable
Self-verifying
Timely / Transparent

Tests doubles

Dummy	Not make anything
Stubs	Canned responses
Spies	Save information
Mocks	Expectations
Fake	Near to real one

Tests doubles

Dummy Not make anything

```
class DummyAuthorizer implements Authorizer {

    public function authorize(string $username, string $password)
    {
        return null;
    }
}
```

Tests doubles

Stub Canned responses

```
class DummyAuthorizer implements Authorizer {

    public function authorize(string $username, string $password)
    {
        return true;
    }
}
```

Tests doubles

Spies Save information

```
class AcceptingAuthorizerSpy implements Authorizer {

	public $authorizerWasCalled = false;

	public function authorize(string $username, string $password)
	{
		$this->authorizerWasCalled = true;
		return true;
	}
}
```

Tests doubles

Mocks Expectations

```
class AcceptingAuthorizerVerificationMock implements Authorizer {

	public $authorizerWasCalled = false;

	public function authorize(string $username, string $password)
	{
		$this->authorizerWasCalled = true;
		return true;
	}

	public function verify()
	{
		return $this->authorizerWasCalled;
	}
}
```

Tests doubles

Fake Near to real one

```
class AcceptingAuthorizerFake implements Authorizer {

	public function authorize(string $username, string $password)
	{
		return $username === "John";
	}

}
```

How do I improve my tests?

1

One assertion per test

```
public function testMultipleOfThreeReturnFizz ()
{
    $this->assertEquals($this->fizzBuzz->run(3), FizzBuzz::FIZZ);
}
```

2

Expect literals

```
public function testMultipleOfThreeReturnFizz() {
      $this->assertEquals( $this->fizzBuzz->run( 3 ), 'Fizz' );
}
```

Do not test:

- Language features
- Standard library
- Frameworks features or classes
- Your stubs

How do you write hard to test code?

How do you write hard to test code?

Work in a constructor

How do you write hard to test code?

Global state
Static methods
Singletons

How do you write hard to test code?

Deep inheritance

How do you write hard to test code?

Non deterministic code

How do you write hard to test code?

Too many conditionals

How do you write hard to test code?

Long methods?
Private variables?

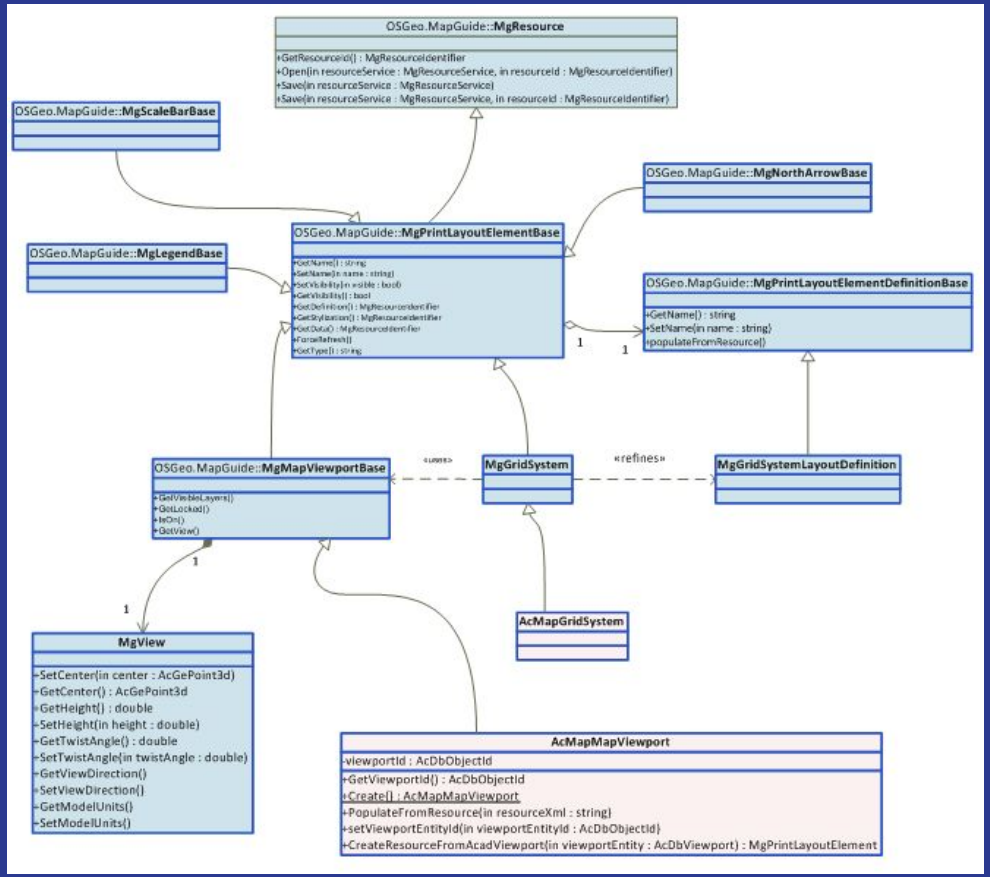
How do I write testable code?

How do I write testable code?

Good OO
Dependency injection
Test Driven Development

How do I write testable code?

Dependency injection


OSGeo.MapGuide::MgResource
+GetResourceId() : MgResourceIdentifier
+Open(in resourceService : MgResourceService, in resourceId : MgResourceIdentifier)
+Save(in resourceService : MgResourceService)
+Save(in resourceService : MgResourceService, in resourceId : MgResourceIdentifier)
OSGeo.MapGuide::MgScaleBarBase
OSGeo.MapGuide::MgNorthArrowBase
OSGeo.MapGuide::MgLegendBase
OSGeo.MapGuide::MgPrintLayoutElementBase
+GetName() : string
+SetName(in name : string)
+SetVisibility(in visible : bool)
+GetVisibility() : bool
+GetDefinition() : MgResourceIdentifier
+GetStyle(in) : MgResourceIdentifier
+GetData() : MgResourceIdentifier
+ForceRefresh()
+GetType() : string
OSGeo.MapGuide::MgPrintLayoutElementDefinitionBase
+GetName() : string
+SetName(in name : string)
+populateFromResource()
1
1
OSGeo.MapGuide::MgMapViewportBase
+GetVisibleLayers()
+GetLocked()
+IsOn()
+GetView()
«uses»
MgGridSystem
«refines»
MgGridSystemLayoutDefinition
1
1
MgView
+SetCenter(in center : AcGePoint3d)
+GetCenter() : AcGePoint3d
+GetHeight() : double
+SetHeight(in height : double)
+GetTwistAngle() : double
+SetTwistAngle(in twistAngle : double)
+GetViewDirection()
+SetViewDirection()
+GetModelUnits()
+SetModelUnits()
AcMapGridSystem
AcMapMapViewport
-viewportId : AcDbObjectId
+GetViewportId() : AcDbObjectId
+Create() : AcMapMapViewport
+PopulateFromResource(in resourceXml : string)
+setViewportEntityId(in viewportEntityId : AcDbObjectId)
+CreateResourceFromAcadViewport(in viewportEntity : AcDbViewport) : MgPrintLayoutElement

Arrange

Act

Assert

Given

When

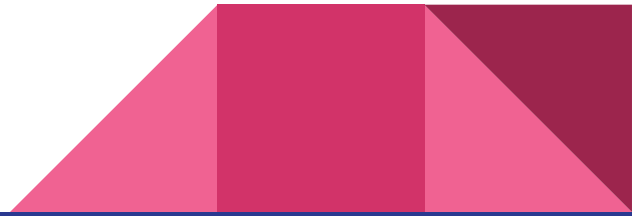
Then

4 rules of simple design

1. Run all the tests
2. Reveals all the intention
3. No duplication
4. Fewest number of classes or methods

The key is to test the areas that you are most worried about going wrong. That way you get the most benefit of your testing effort.

Martin Fowler



Practise makes perfect

Thank you