Marcin Rybak's Home

ADF BC in Fusion trainings

- Fusion database model essentials
- ADF Entity Objects in Fusion essentials
- ADF View Objects in Fusion essentials
- ADF Application Modules in Fusion essentials
- ADF training - live coding

ADF REST

- ADF Rest framework demystified
- How to create ADF REST service

ADF/Fusion - Other

- ADF good practices - part 1
- ADF Task flows essentials
- Effective ADF UI development techniques
- How to fix Fusion audit violations
- How to fix common issues
- ADE How-To

Other

- Mocking in Java
- Find a catchy name for the Container application

Recently Updated

Marcin Rybak's Home
36 minutes ago • updated by Marcin Rybak • view change

Public documents
about 10 hours ago • updated by Marcin Rybak • view change

ADF training - live coding
Sep 06, 2016 • updated by Marcin Rybak • view change

ADF-training.png
Sep 06, 2016 • attached by Marcin Rybak

ADF-training
Sep 06, 2016 • attached by Marcin Rybak

Volunteering - refactorings
Jul 15, 2016 • updated by Marcin Rybak • view change

Research
Apr 16, 2016 • updated by Marcin Rybak • view change

Mocking in Java
Apr 07, 2016 • updated by Marcin Rybak • view change

Find a catchy name for the Container application
Feb 11, 2016 • updated by Marcin Rybak • view change

Drawing2.xml
Feb 11, 2016 • attached by Marcin Rybak

Drawing2.xml.png
Feb 11, 2016 • attached by Marcin Rybak

Private documents
Feb 10, 2016 • created by Marcin Rybak

Drawing1.xml
Feb 10, 2016 • attached by Marcin Rybak

Drawing1.xml.png
Feb 10, 2016 • attached by Marcin Rybak

Drafts
Jan 26, 2016 • created by Marcin Rybak

Private documents

confluence mastering

FDR template

Name	Date

Requirements template

Target release	
Epic	
Document status	**DRAFT**
Document owner	Marcin Rybak
Designer	
Developers	
QA	

Goals

Background and strategic fit

Assumptions

Requirements

#	Title	User Story	Importance	Notes
1				

User interaction and design

Questions

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

Not Doing

test descision template

Status	**NOT STARTED**
Stakeholders	
Outcome	
Due date	01 Jan 2016
Owner	Marcin Rybak

Background

test desc

Action items

- [ ]

Drafts

Research

Fusion APIs

Functionality	Library Name	Library Path	Module name	Usage

Fetch country specific 1st level administrative division name
 	
(depends on particular country, i.e state, wojewodztwo)
	AdfFoundationGeographiesPublicView	fusionapps/jlib	GeoStructureLevelPVO	`ViewObjectImpl vo = get`
				`vo.appendViewCriteria(v`
				`vo.setNamedWhereClauseP`
				`vo.executeQuery();`

Fetch country 1st level administrative divisions codes list	AdfFoundationGeographiesPublicView	fusionapps/jlib	GeographyElement2PVO	`ViewObjectImpl vo = get`
				`vo.appendViewCriteria(v`
				`vo.setNamedWhereClauseP`
				`vo.setNamedWhereClauseP`
				`vo.executeQuery();`

Fetch administrative divisions display names	AdfFoundationGeographiesPublicView	fusionapps/jlib	GeographyIdentifierPVO	*Standard use*
Build custom queries which operate on Fusion's Geographic data tables	AdfFoundationGeographiesPublicEntity	fusionapps/jlib	• GeographyPEO • GeoStructureLevelPEO • GeographyIdentifierPEO	Build one PVO which returns the 1st level administrat

Public documents

ADF training - live coding

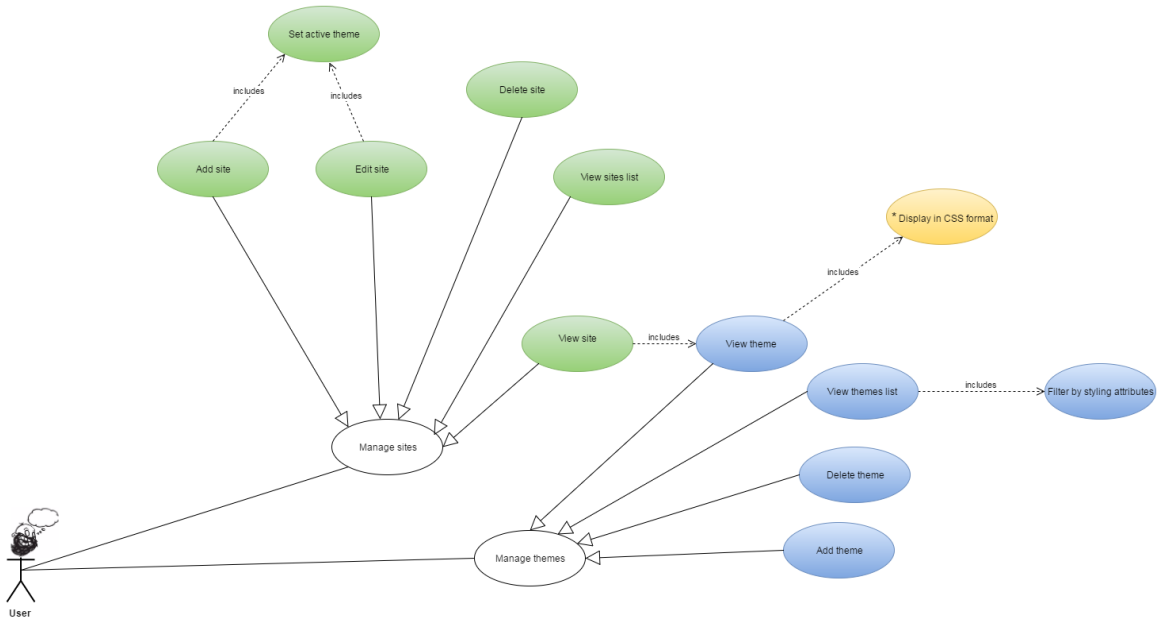
Description

Create an ADF Fusion application on R13 techstack, which allows to configure styling for web sites, following ADF Fusion standards and good practices.

Involved areas

- database
- ADF BC layer

Supported use cases



Add site

In order to manage web sites in an Application

As an User

I want to add a new site

Scenario 1 (basic)	*Scenario 2 (required fields validation)*	*Scenario 3 (site code uniquess)*
Given at least one theme is defined in Application **And** User properly sets site code, site name for current session's language and selects active theme **When** User commits the changes **Then** the data should be persisted and new site **visible** on view sites screen	**Given** User adds a site entry but doesn't type in site code or site name or select active theme **When** User commits the changes **Then** validation should indicate that required information is missing and should keep user on the edit page	**Given** User adds a site entry and filled site code that already exists **When** User commits the changes **Then** validation should indicate that code is not unique and should keep User on the edit page

Delete site

In order to manage web sites in an Application

As an User

I want to delete site

View sites list

In order to manage web sites in an Application

As an User

I want to display all web sites list, including attributes like site code and site name for current session's language

View site

In order to manage web sites in an Application

As an User

I want to display any web site attributes like site code and site name for current session's language as well as all active theme attributes

View theme

In order to manage web sites themes in an Application

As an User

I want to display any web site theme attributes like theme code, background-color, font-size, font-family and

**generated CSS, based on these attributes*

View themes list

In order to manage web sites themes in an Application

As an User

I want to display all web site themes list

Scenario 1 (basic)	Scenario 2 (filter by styling attributes)
Given some site themes were already **entered**	**Given** some site themes were already **entered**
When User accesses view site themes screen	**When** User accesses view site themes screen
Then all previously **entered** site theme entries should be visible, including attributes like theme code, background-color, font-size and font-family	**Then** User should be able to filter all previously **entered** site theme entries by background-color, font-size and font-family, providing the fragments of these attributes values

Delete theme

In order to manage web site themes in an Application

As an User

I want to delete site theme

Scenario 1 (basic)	Scenario 2 (delete active theme)
Given some site themes were already **entered** **When** User deletes the chosen site theme **Then** the entry should be removed from persistence layer and no longer **visible** on view site themes screen	**Given** some site themes were already **entered** **When** User deletes the theme which is selected as an active theme for any site **Then** the entry should be NOT removed from persistence layer and an error message should be displayed

Add theme

In order to manage web site themes in an Application and assign them to any web site

As an User

I want to add a new site theme

Scenario 1 (basic)	*Scenario 2 (required fields validation)*	*Scenario 3 (site theme code uniquess)*
Given User properly sets site theme code, background-color, font-size and font-family **When** User commits the changes **Then** the data should be persisted and new site **visible** on view site themes screen	**Given** User adds a site theme entry but doesn't type in site theme code or background-color or font-size or font-family **When** User commits the changes **Then** validation should indicate that required information is missing and should keep User on the edit page	**Given** User adds a site theme entry and filled site theme code that already exists **When** User commits the changes **Then** validation should indicate that code is not unique and should keep User on the edit page

Technical solution action items

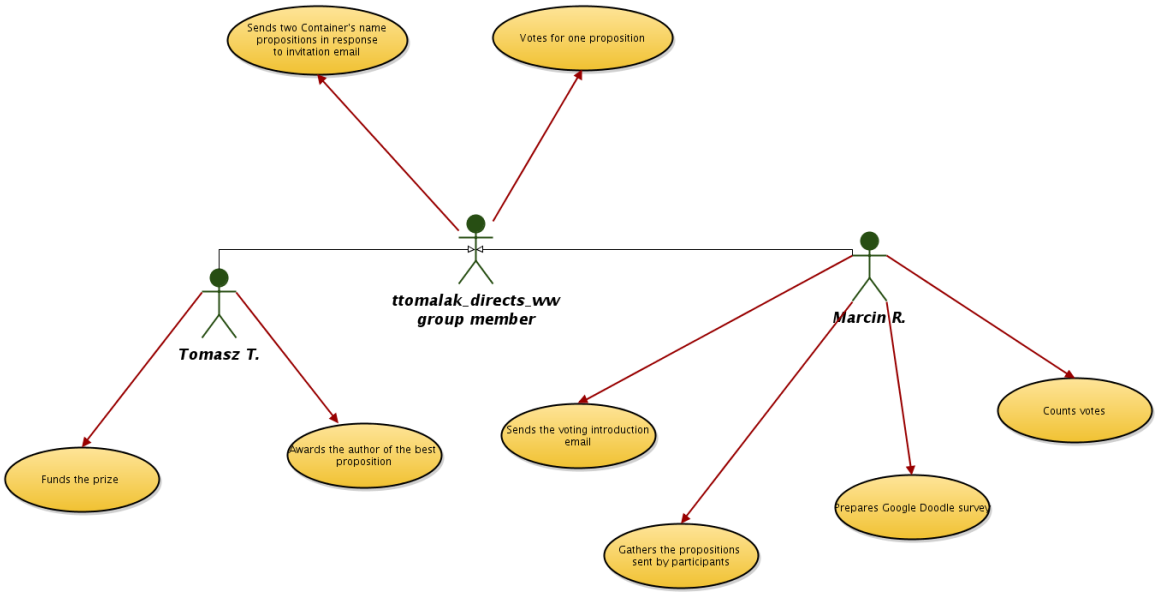
Action
Create ADF Fusion Application on R13 stack
Create Offline DB object project and appropriate offline DB objects
Deploy Offline DB objects on a database
Create protected-scope model project and appropriate ADF BC
**Create public-scope view object project containing read only view object returning site themes list in CSS format (SELECT THEME_CODE, CSS FROM ...)*

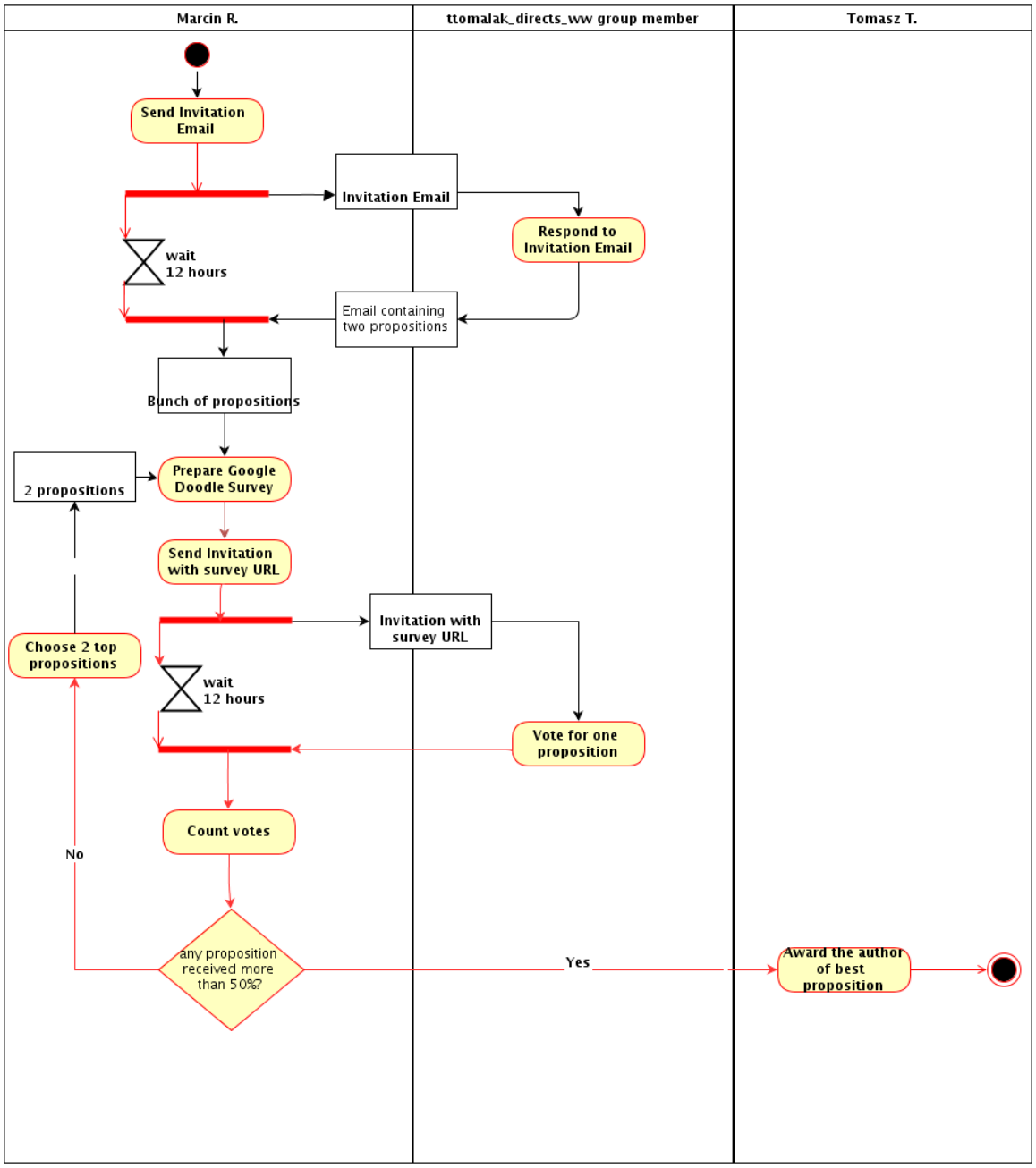
Acceptance criteria

Criterion	Priority
All technical solution action items were followed	
All use cases are covered - demonstrate with ADF BC Tester	
DM validation on a database for created model shows no violations	

jaudit validation on created ADF BC shows no violations	

Find a catchy name for the Container application





Mocking in Java

What is mocking?

Collins Dictionary

mock

verb

1. when intr, often foll by at to behave with scorn or contempt (towards); show ridicule (for)
2. (transitive) **to imitate,** esp in fun**; mimic**
3. (transitive) to deceive, disappoint, or delude
4. (transitive) to defy or frustrate the team mocked the visitors' attempt to score

noun

1. **the act of mocking**
2. a person or **thing mocked**
3. **a counterfeit; imitation**
4. (often plural) (informal) (in England and Wales) the school examinations taken as practice before public examinations

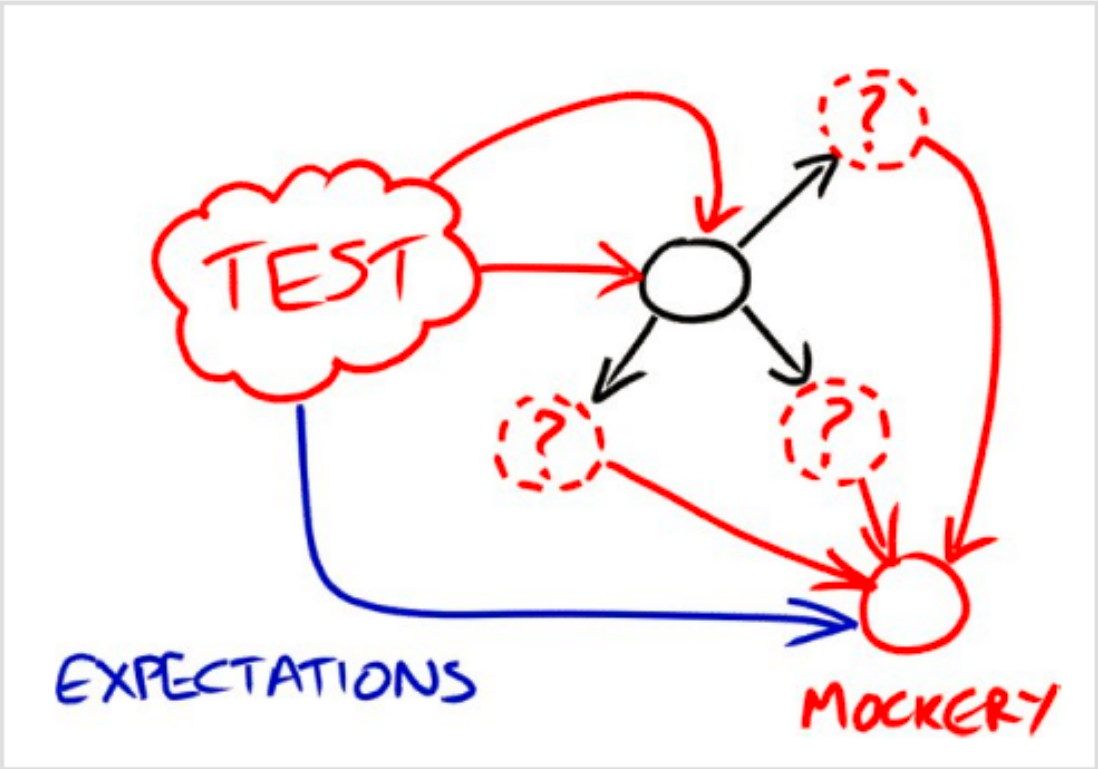
adjective (prenominal)

1. sham or counterfeit
2. serving as an imitation or substitute, esp for practice purposes a mock battle, mock finals

> ***Mocking means imitating....***

Steve Freeman in "TDD 10 years later" :

You fire something at a particle, things splinter off and you can detect what happens…



> ***...but also mocking means veryfying behaviour***

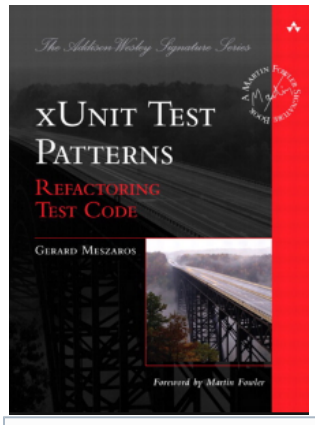
Mocks, stubs et al. ...

What are test doubles?

Collins Dictionary

stunt double
noun

1. (cinema) someone who performs dangerous stunts in a film in place of an actor, See also stuntman, stuntwoman



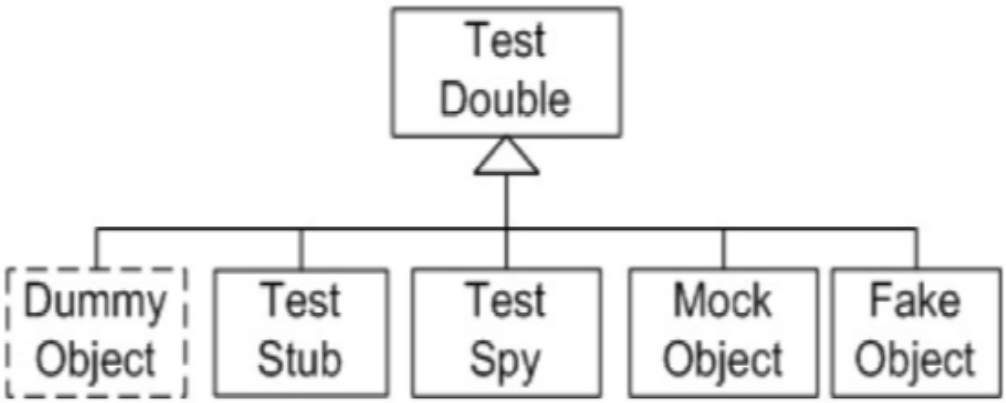
From "XUnit Test patterns" (http://xunitpatterns.com/):

A general name for objects used to replace real components for testing purposes, like:

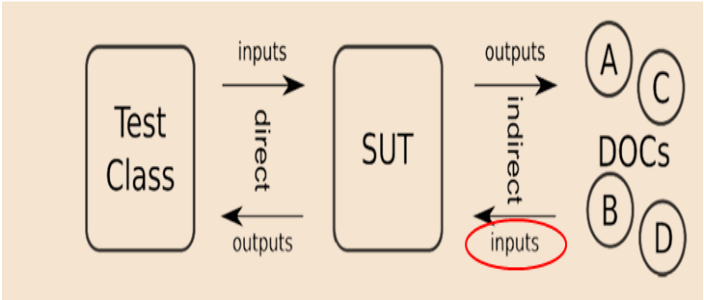
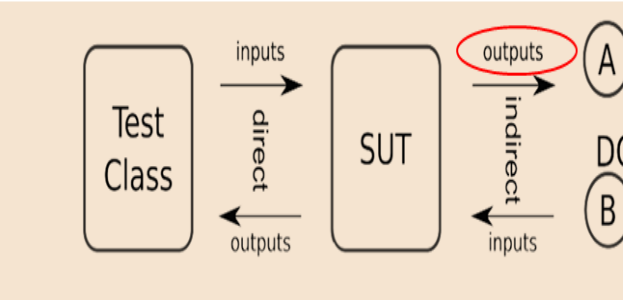
- Dummy

- Fake
- Stub
- Mock
- Spy

Test doubles systematics

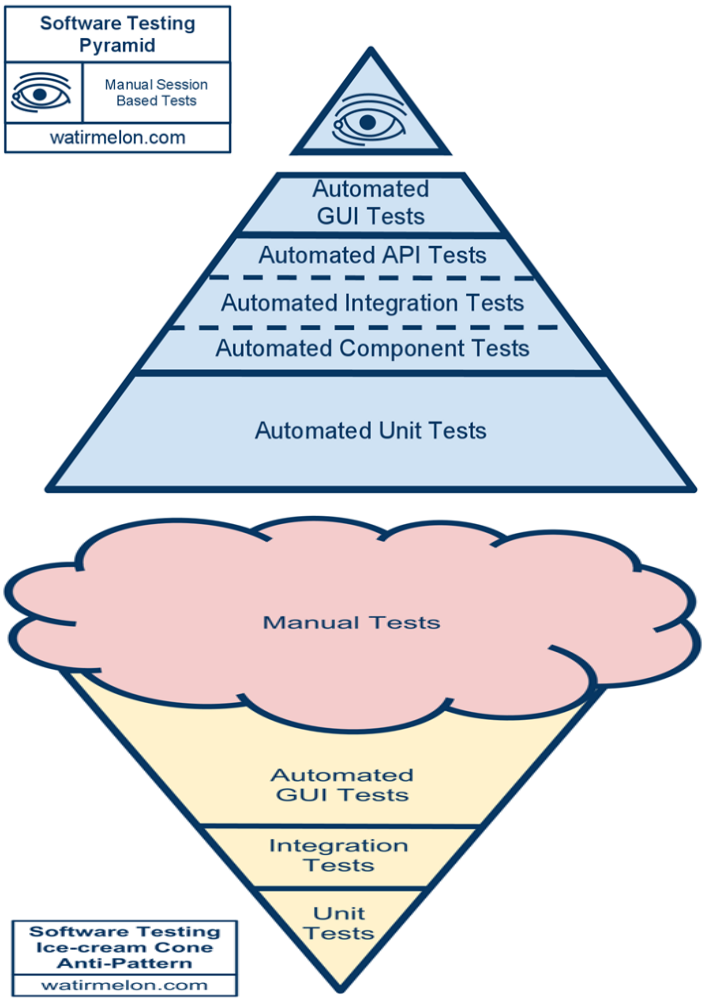


Dummy	Fake
• Just needs to exist, no real collaboration needed • Make the code compile and executable	• Real implementation, but simpler, i.e memory data base • Cheaper replacement of real object • Mainly used in integration tests

Stub	Mock
• Used for passing some values to the SUT (“System Under Test indirect inputs”) • Returns a hard coded answer • The Stub has the same interface as the real class, but has no logic or advanced behaviour • State verification from SUT	• A Mock is like a Stub, but additionally it verifies interactions. • Used to verify if the SUT calls specific methods of the collabor (“indirect outputs”) • Can be partial - if so it is a **spy**
	

Why to mock?

To design the code	To test the code

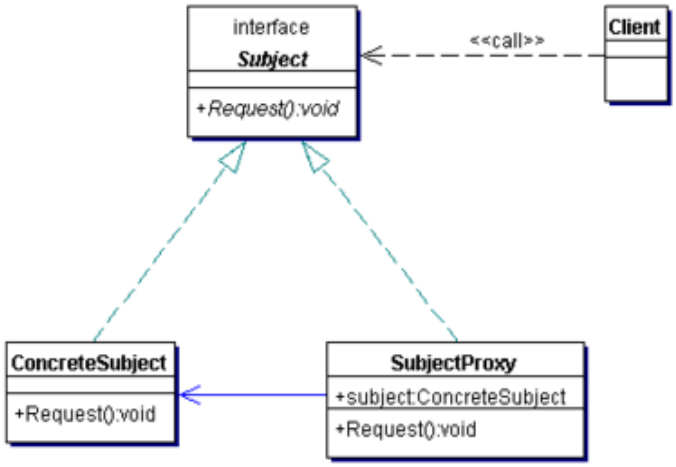
• Mocking supports TDD
• It is like doing a UML sequence diagram, but in code
• Mocking speeds up development - collaborating modules can be written parallelly
• Inability to mock code under test dependencies reveal design flaws (i.e. to much coupling)
and drives the code refactoring and improvement
	
	• Mocking supports testing code without testing the dependencies
	• simulate the behavior of complex, real objects when:
	  • real object has states that are difficult to create or reproduce (e.g., a network error)
	  • real object supplies non-deterministic results (e.g., the current time or the current temperature)
	  • real object is slow (e.g., a complete database, which would have to be initialized before the test)

How does it work?

- Mock objects have the same interface as the real objects they mimic, allowing a client object to remain unaware of whether it is using a real object or a mock object

Dynamic proxies approach

- Jmock, Mockito
- A proxy object is used to imitate the real object your code is dependent on
- A proxy forces object method calls to occur indirectly through the proxy object, which acts as a surrogate or delegate for the underlying objec proxied
- Proxy design pattern - tight coupling:



- Dynamic Proxy API was introduced in Java 1.3 - the core units are: `java.lang.reflect.Proxy` class and `java.lang.reflect.Invoc er` interface:

```
public interface MyCollaborator
{
    public Object doSomething(String param);
}
```

```
public class MyHandler implements java.lang.reflect.InvocationHandler {

  MyCollaborator proxiedObject;

  public MyHandler(MyCollaborator proxiedObject) {
     this.proxiedObject = proxiedObject;
  }

  public Object invoke(Object proxy, Method m, Object[] args) throws Throwable {
    // itercept all method calls on proxy instance - do stubbing, verify
expectations
  }
}
```

```
MyCollaborator proxyInstance= (MyCollaborator)
java.lang.reflect.Proxy.newProxyInstance(MyCollaborator.class.getClassLoader(),
                                         Class[] { MyCollaborator.class },
                                         new MyHandler(new MyCollaborator(){...}
Object res = proxyInstance.doSomething("ble");
```

- A *dynamic proxy class* implements a list of interfaces specified at runtime when the class is created and extends `java.lang.reflect.Proxy` class
- Each proxy instance has an associated *invocation handler* object, which implements the interface `InvocationHandler`.
- A method invocation on a proxy instance will be dispatched to the `invoke` method of the instance's invocation handler, passing:
  - the proxy instance,
  - `java.lang.reflect.Method` object identifying the method that was invoked,
  - an array of type `Object` containing the arguments.
- The invocation handler processes the encoded method invocation as appropriate and the result that it returns will be returned as the result of the method invocation on the proxy instance.
- There are a few important restrictions to the proxies. It's not possible to:
  - intercept static method calls
  - intercept private method calls
  - intercept final method calls
  - build a proxy for a final class

> When no collaborators interfaces are available **CGLib proxies** can be used
>
> (CGLib is a third-party framework, based on bytecode manipulation).

Why not to mock?

Some anonymous mocking antagonist:

Mocks are like hard drugs... the more you use, the more separated from reality everything becomes.

Steve Freeman:

No tool nor technique can survive inadequately trained developers.

- Mocking decreases tests readability and simplicity

  Example
  Instead of just a straightforward usage of the code (e.g. passing in some values to the method under test and checking the return result), extra code needs to be written to tell the mocks how to behave. Having this extra code detracts from the actual intent of what is being tested, and very often this code is hard to understand.

- High cost of tests maintenance when the use of mock objects closely couple the unit tests to the actual implementation of the code that is being tested

  Example
  Many mock object frameworks allow the developer to check the order of and number of times that mock object methods were invoked by the real object being tested; subsequent refactoring of the code that is being tested could therefore cause the test to fail even though all mocked object methods still obey the contract of the previous implementation. Unit tests should test a method's external behavior rather than its internal implementation.

  If you're trying to read a test that uses mocks and find yourself mentally stepping through the code being tested in order to understand the test, then you're probably overusing mocks

- Test-driven design changes may pollute the public interface of the tested object.

  Example
  Forcing the public API to accommodate factory objects or abstract factories, just because the test code wants it to, is the wrong design. If production code has no need to instantiate many types of this collaborator, then adding that ability will make the resulting design needlessly hard to understand.

  Protected factory method which returns this collaborator instance is the right solution here.

- If the behavior of mocked real object is not modeled correctly then the unit tests may register a pass even though a failure would occur

  Example
  When you tell a mock how to behave, the only assurance you get with your tests is that your code will work if your mocks behave exactly like your real implementations. This can be very hard to guarantee, and the problem gets worse as your code

changes over time, as the behavior of the real implementations is likely to get out of sync with your mocks.

- Do not use mocks when what you really need is an integration test

Example
Sometimes you can't use a real dependency in a test (e.g. if it's too slow or talks over the network), but there may better options than using mocks, such as a hermetic local server (server that you start up on your machine specifically for the test) or a fake implementation (e.g. an in-memory server).

For More information about using hermetic servers, see http://googletesting.blogspot.com/2012/10/hermetic-servers.html.

Mocking frameworks

	Jmock
History	• Fist version in 2000 • Created by Steve Freeman, Tim Mackinnon, Nat Pryce et al. • Current version **2.6.0** (Java 6) from 2012

Philosophy

1. A Good Programmer does Language Design.

> *"[...] a good programmer in these times does not just write programs. [...] a good programmer does language design, though not from scratch, but building on the frame of a base language."*
> *— Guy Steele Jr. [13]*

Every program is a new language. That language may be confused and implicit, but at a minimum there will be conventions and programming interfaces that color the structure of the code. The art of writing software well is to tease out the concepts in a domain and make them concrete and tractable, to make the language within the program explicit.

jMock has evolved over several years from a primitive class library into a more complex framework. The driving force for the change was the need for clearer and more powerful specification of the expectations. As jMock evolved, its API changed from being an object-oriented library into what we now understand to be an embedded domain-specific language. The domain of that language is the specification of how objects should interact within a test scenario and the interpreter of the language is the testing framework itself.

The following sections describe the evolution of jMock, illustrating the forces that led to the development of an EDSL. To stretch a metaphor, we arrived at the current design through several generations of evolution. As we struggled with the limits of each implementation, the environment changed and we moved to new designs that were more effective. Curiously, some of our rejected designs survive in frameworks that were developed in isolation from the original team.

Creating Mocks

```
import org.jmock.Mockery;

Mockery context = new Mockery();
Collaborator collaborator = context.mock(Collaborator.class);
```

```
@RunWith(JMock.class)
public class Test {
    @Mock
    private Collaborator collaborator
```

Mocking Classes	`Mockery context = new Mockery() { {`
	`setImposteriser(ClassImposteriser.INSTANCE);`
	`} };`

Same class mocked twice	`Collaborator collaborator1 = context.mock(Collaborator.class,"col1");` `Collaborator collaborator2 = context.mock(Collaborator.class,"col2");`

Expectations	`Mockery context = new Mockery();`
	`Mock collaborator = context.mock(Collaborator.class);`
	
	`collaborator.expects(once()).method("getSomething").with(eq("some para`
	`.will(returnValue("some result"));`
	`//call code which depends on collaborator -> use collaborator.proxy()`
	
	`Mockery context = new Mockery();`
	`final Collaborator collaborator = context.mock(Collaborator.class);`
	
	`context.checking(new Expectations() { {`
	`oneOf(collaborator).getSomething("some param");`
	`will(returnValue("some result"));`
	`}}`
	`);`
	
	`//call code which refers collaborator`

Argument Matchers	`any(Integer.class),`
	`any(String.class);`
	`with(100);`

Number of Invocations	`new Expectations(){{`
	` exactly(1).of(collaborator);`
	` atLeast(1).of(collaborator);`
	` never(collaborator);`
	` }}`
	`);`

Consecutive Calls	`new Expectations() { {`
	`oneOf(collaborator).getSomething(with(100), with(any(Strir`
	`will(onConsecutiveCalls(returnValue(true), returnValue(fal`
	`returnValue(false)));`
	`} }`
	`);`

Verify Call Order	`final Sequence sequence = context.sequence("order");`
	`context.checking(new Expectations() { {`
	`oneOf(collaborator).someMethod(with(any(Integer.class)),with(any(Str`
	
	`will(returnValue(true));`
	`inSequence(sequence);`
	
	`atLeast(1).of(collaborator2).someMethod(with(100), with(any(String.c`
	`will(returnValue(true));`
	`inSequence(sequence);`
	`} }`
	`);`

General Impressions after working	• Bases on expectations which tie stubbing and verification together • By default, JMock assumes that a test double (a “mock”) expects clients not to invoke anything at any time. To relax that as to be added. • JMock encourages to clarify the interaction that are needed and stops from introducing dependencies recklessly. • JMock tries its best to stop from introducing accidental complexity.
When to use?	• Supports designing new features

Detroit vs London

	London	Detroit
Religion	Mockists	Classicists
Genesis - When and where	1999 - XP adopters in London	1996 - C3 project in Detroit
Gurus	Steve Freeman, Nat Pryce	Kent Beck, Uncle Bob
Credo	Always use a mock for any object with interesting behavior.	Use real objects if possible and a double if it's awkward to use the real thing

> **Uncle Bob says:**
>
> (...) And write your own mocks. Try to depend on mocking tools as little as possible. Or, if you decide to use a mocking tool, *use it with a very light touch*.

References

- http://www.jmock.org/oopsla2006.pdf
- http://monkeyisland.pl/2008/01/14/mockito/
- http://zsoltfabok.com/blog/2010/08/jmock-versus-mockito/
- http://blog.thecodewhisperer.com/permalink/jmock-v-mockito-but-not-to-the-death/
- http://www.michaelminella.com/testing/the-concept-of-mocking.html
- http://media.pragprog.com/articles/may_02_mock.pdf
- http://www.ibm.com/developerworks/java/library/j-mocktest/index.html
- https://benbiddington.wordpress.com/tag/mocking/
- http://www.mockobjects.com/2009/09/brief-history-of-mock-objects.html
- http://googletesting.blogspot.ro/2013/05/testing-on-toilet-dont-overuse-mocks.html
- http://martinfowler.com/articles/mocksArentStubs.html
- http://www.slideshare.net/intellizhang/tdd-and-mock-objects
- https://blog.8thlight.com/uncle-bob/2014/05/10/WhenToMock.html
- http://www.slideshare.net/davidvoelkel/mockist-vs-classicists-tdd-57218553
- https://gojko.net/2009/09/21/mocks-are-not-about-isolation-but-about-responsibilities/
- http://www.javaworld.com/article/2076233/java-se/explore-the-dynamic-proxy-api.html
- http://blog.javabenchmark.org/2013/05/java-instrumentation-tutorial.html

Volunteering - refactorings

Area	Detailed description	Priority	Action Required	Effort	Status
Create New Project	Invalid packages (beans, flow, pages)	High	Rename packages according to UI Folder Structure and Naming Conventions	1mD	PENDING
					IN PROGRESS
					DONE