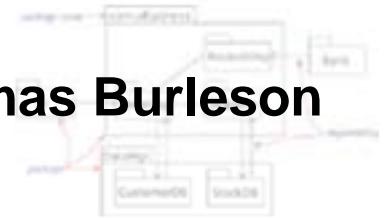




Thomas Burleson



**PRINCIPAL ARCHITECT
CERTIFIED INSTRUCTOR**
Adobe® Flex™



ThomasB@UniversalMind.com

<http://www.thomasburleson.biz/>

Continuous Testing

with

Flex & Cairngorm

Source code samples

@ blog.universalmind.com

Oooh no!

Not another session of "how to test"...

DISCLAIMER

This is a 2-day course.... Condensed into 1 hour.

Prepare to FLY!

Let us explore some concepts!

What are the differences ?

Is this done on the server or on each dev box ?

Is this for UI or Non-UI ?



Unit Testing:

Test the functionality and stability of a single unit;

Test ALL the functionality of the unit

e.g. `PersonnelDOA.cfc` or `PersonnelCommand.as`



Continuous Testing:

Testing all units during each cycle (svn commit or build)

Gather errors or results for easy review or automated reporting

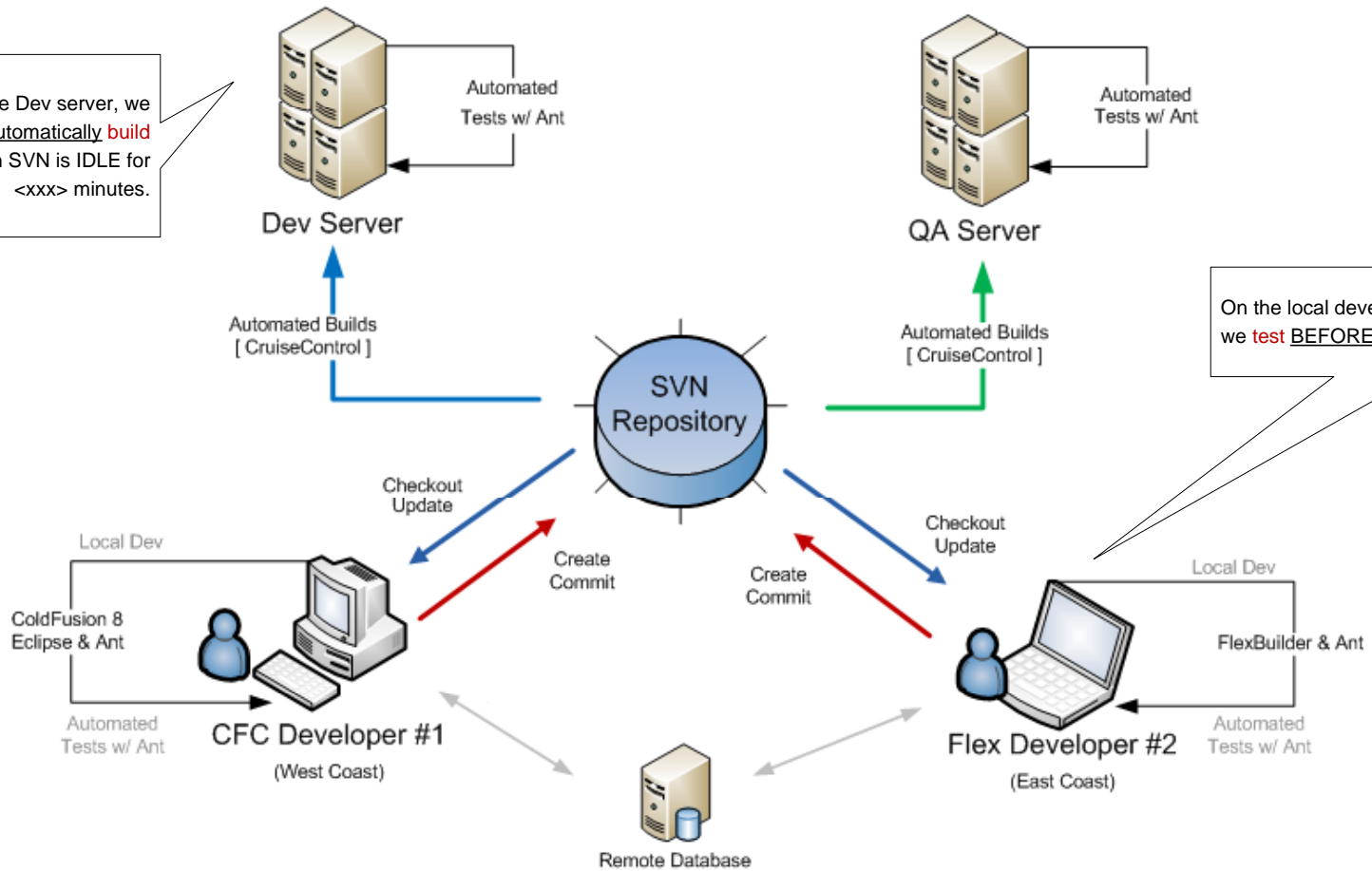


Continuous Integration:

Integrate with other developers test modules for “complete” integrated testing of all units

Usually we are discussing the “server processes” here...

On the Dev server, we continuously & automatically build and test when SVN is IDLE for <xxx> minutes.



On the local developer machine, we test BEFORE SVN check-in

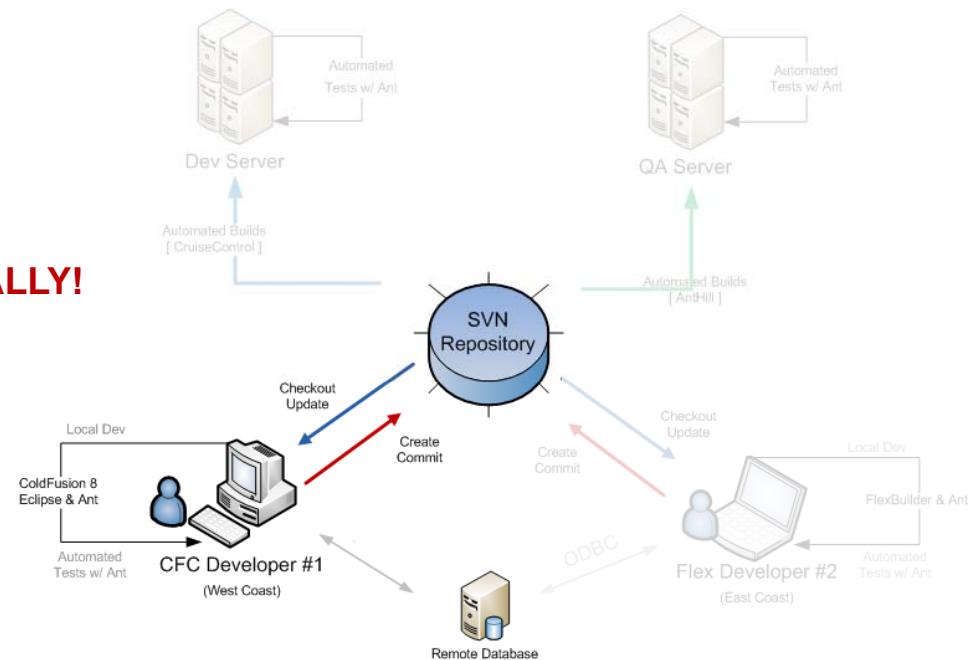
So **why** this session “Continuous Testing”?



Improve your development style

Improve quality of Unit Testing

Perform continuous testing... **LOCALLY!**



Testing Flex Applications

Using **Cairngorm** and **FlexUnit** Testing

Testing **Flex** applications with **FlexUnit**

What are we testing?

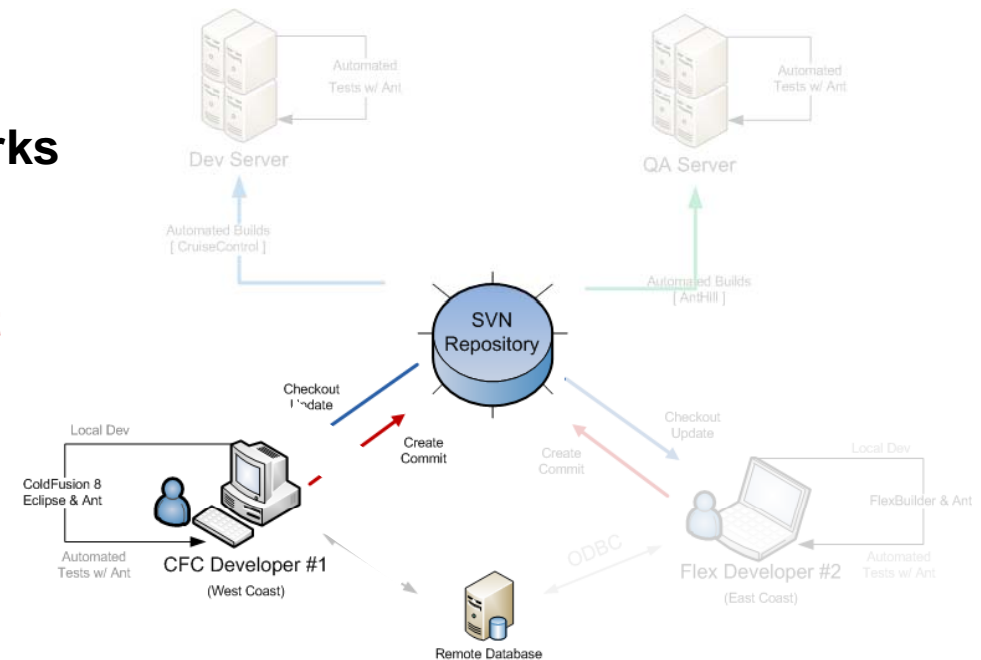
1. Testing API from RDS server-tier
2. Testing transformations from Delegates
3. Testing business logic
4. Testing event processing
5. Testing model updates

This is **NOT** testing UI; that is functional testing...

- Announce user gesture by dispatching events (includes databindings)
- UI updating in response to model changes

Overview

- 1) Traditional Approach [Test Shells]
- 2) Using Testing Framework(s)
- 3) **Misconceptions**
- 4) **Problems** with Testing Frameworks
- 5) Testing with **Cairngorm** MVC
- 6) Using extensions to UM **FlexUnit**
- 7) Sample code



Traditional Approach

Step 1:

Create 1 or more MXML or AS classes
[UIComponent descendants ? Yes or No]

Step 2:

Create 1 or more Flex Test Apps to **test** & **debug** the Flex Views
Create 1 or more Flex Test Apps to test & debug **non-UIComponent** code.



This “traditional” approach is the same even if using web
MVC frameworks such as **Mach-ii** or **Struts**.



Test_RelatedCompanies. (mxml Application)

1. Must be run manually.
2. Does not related to MVC easily.
3. Is a very narrow slice of UI functionality.
4. see next slide...

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
3
4   <mx:Script>
5     <![CDATA[
6
7       private function onRelatedStockSelected(event:StockSelectedEvent):void { trace(event.stockID); }
8
9       [Bindable] private var __relatedCompanies : Array = MockCompanies.buildRelatedCompanies();
10      [Bindable] private var __storm : StormVO = new StormVO();
11
12    ]]>
13  </mx:Script>
14
15  <news:RelatedStockCodes
16    id="cwTitle"
17    width="100%"
18    height="185"
19    stormID="{ __storm.stormId }"
20    relatedCompanies="{ __storm.relatedCompanies }"
21    showLoading="{false}"
22    title="Related Companies"
23    helpImage="{ helpIcon }"
24    stockCodeSelected="{ onRelatedStockSelected(event); }"
25    helpToolTip="Related Companies determines from..."
26    xmlns:news="com.clientX.demo.view.stock.*"/>
27
28  <!--
29    REAL Version
30  <news:RelatedStockCodes
31    id="cwTitle"
32    width="100%"
33    height="185"
34    stormID="{ __model.storm.stormId }"
35    relatedCompanies="{ __model.storm.relatedCompanies }"
36    showLoading="{ __model.storm.loadingRelatedCompanies}"
37    title="Related Companies"
38    helpImage="{ helpIcon }"
39    stockCodeSelected="{ onRelatedStockSelected(event); }"
40    xmlns:news="com.clientX.demo.view.stock.*"/>
41  -->
42
43 </mx:Application>
44
  
```

Issues with this approach

- Tests are performed outside the MVC framework
 - Data loads are manual or are mocked in an obvious way
 - Events from component(s) are not tested for impacts to model, etc.
- Forces creation of many test shells.
 - This is great for functional UI testing without parent view overhead...
- Developers should differentiate between functional UI testing vs. **unit testing**
 - Functional UI testing is a “human” issue... [or use QTP]
 - Unit testing is a “tool” issue...

Issues with Traditional Approach

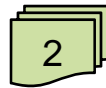
It is Ad-HOC (makeshift...)!



How do make sure you test each method of an Flex class?
... and no way to easily test all classes



Traditional does not promote tests of “**boundary conditions**”.



Traditional cannot build up a **suite of tests** to verify your code.



Traditional provides no easy to perform regression testing of all public functions for every CFC



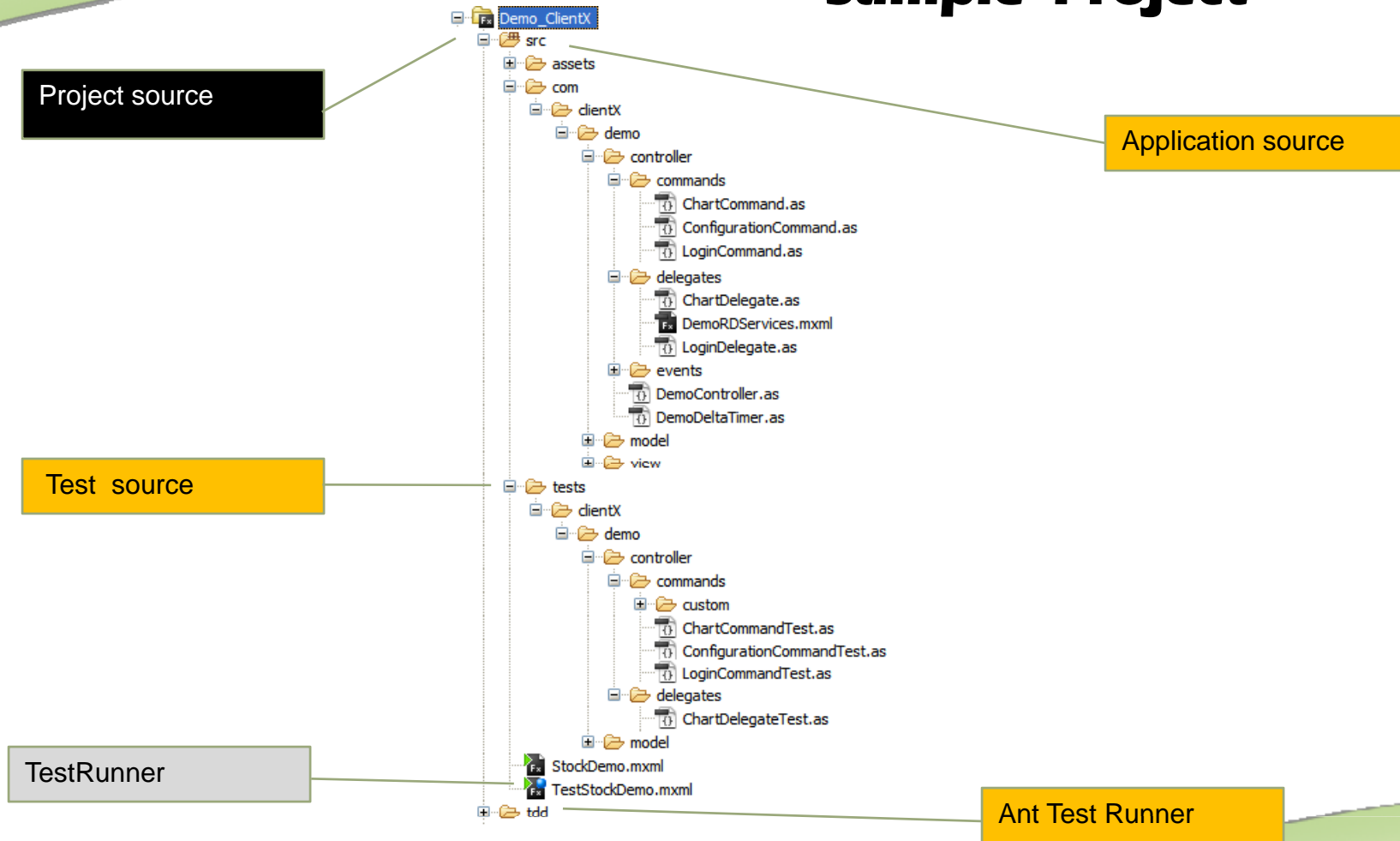
Easy to confuse UI testing as business testing.

How to test within the context of an MVC framework?

1st review **package structures** [with MVC Flex apps]

- Here are some common questions:
 1. How do “real” packages and source relate to test source?
 2. Should everything be tested?
 3. How to test UI?

Sample Project



Test Mappings

- ❖ 1-to-1 mappings of class names
- ❖ Use <xxxx>Test notations

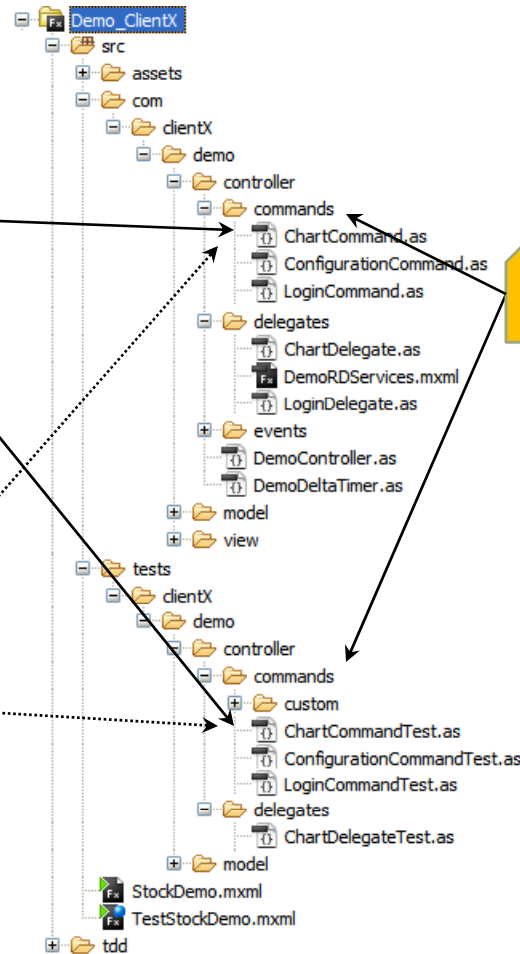
ChartCommand.as
ChartCommandTest.as

- ❖ 1-to-1 mappings function names
- ❖ Use test<xxxx> notations
- ❖ Applies only to public methods

public function getDetailsByCode(x)"
public function testGetDetailsByCode(x)"

- ❖ 1-to-1 mappings of package names

com..commands.*
test...commands.*



Running TestRunner

Use Webserver:

<http://localhost:9600/FlexBootCamp/TestStockDemo.html>

Use Ant:

[Run as Ant Build <projectdir>/tdd/runTests.xml](#)

Tests	Errors	Failures
1	1	0

Success

Tests	Errors	Failures
1	0	0

Execution Time: 1032 ms

```

6. E:\web\JRun5\servers\CFUnited\cfusion.ear\cfusion.war\portalApp\tdd\cfunit\net\sourceforge\cfunit\framework\TestCase.cfc:111
7. E:\web\JRun5\servers\CFUnited\cfusion.ear\cfusion.war\portalApp\tdd\cfunit\net\sourceforge\cfunit\framework\TestResult.cfc:116
8. E:\web\JRun5\servers\CFUnited\cfusion.ear\cfusion.war\portalApp\tdd\cfunit\net\sourceforge\cfunit\framework\TestResult.cfc:102
9. E:\web\JRun5\servers\CFUnited\cfusion.ear\cfusion.war\portalApp\tdd\cfunit\net\sourceforge\cfunit\framework\TestCase.cfc:102
10. E:\web\JRun5\servers\CFUnited\cfusion.ear\cfusion.war\portalApp\tdd\cfunit\net\sourceforge\cfunit\framework\TestSuite.cfc:228
11. E:\web\JRun5\servers\CFUnited\cfusion.ear\cfusion.war\portalApp\tdd\cfunit\net\sourceforge\cfunit\framework\TestSuite.cfc:220
12. E:\web\JRun5\servers\CFUnited\cfusion.ear\cfusion.war\portalApp\tdd\cfunit\net\sourceforge\cfunit\framework\TestRunner.cfc:71
13. E:\web\JRun5\servers\CFUnited\cfusion.ear\cfusion.war\portalApp\tdd\runTests.cfm:34
    
```

```

/com/xxx/business/gateway/ZIPCodeLookupDG.cfc:34
/com/xxx/services/ZIPCodeLookupService.cfc:30
/com/xxx/delegate/ZIPCodeLookupDelegate.cfc:27
/test/xxx/delegate/TestZIPCodeLookupDelegate.cfc:10
funit\net\sourceforge\cfunit\framework\TestCase.cfc:166
    
```

Overview of **Cairngorm** MVC

Class [Adobe] Cairngorm

1. Uses MVC pattern
2. Poor support of Webservices initializations
3. Enterprise issues view notifications
4. Modules not supported

Open-source on
Google 12/07.

Extended with **UM Cairngorm**

1. Support for view responders
2. Support for command event dispatching
3. Superior support for delegate queues
4. Support for batch events (parallel and/or sequential)
5. Support for delegate-level data transformations
6. Support for command logic aggregations
7. Mini-MVC apps/modules supported
8. Usable for full non-ui **FlexUnit** testing

Let's review some Cairngorm diagrams....

- Be BRIEF please!

Testing Frameworks

ASUnit - [Flash](#)

FlexUnit - [Flex](#)

dpUnit - [Flex \(Air\)](#)

UM Wrappers - [FlexUnit](#)

Step 1:

Quick Review of FlexUnit

- a) Installation
- b) Creating / Using TestCases

Step 2:

Using TestRunner to run the TestCases

Step 3:

Using Eclipse Ant to run the TestCases

Using **FlexUnit**



Using FlexUnit Framework



Building FlexUnit Testcases



Configure TestRunner w/ TestCases



Launch TestRunner



Incorporate into CruiseControl

Uses Flex GUI [TestRunner]
to execute TestCases!

And those TestCases test
only Non-GUI code...

Common **Misconceptions**



Have to write the test code 1st
... can you say **Test-Driven Development**



Have to change development style



If the code base already exists without tests... Its too late!



Cannot use when fixing a new bug

Step 1: write the test to verify the bug exists

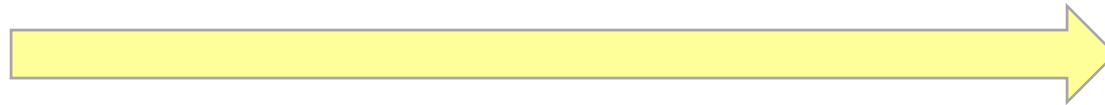
Step 2: fix the bug

Step 3: run the test to verify



Do I have to test everything?

Hey speaker-dude...



Review some real-world source code !

Why are people **resistant** to testing?



I do not have **time** to write test code...



Too much business code **already exists**

... and it is too late to start now & generate test cases for each class



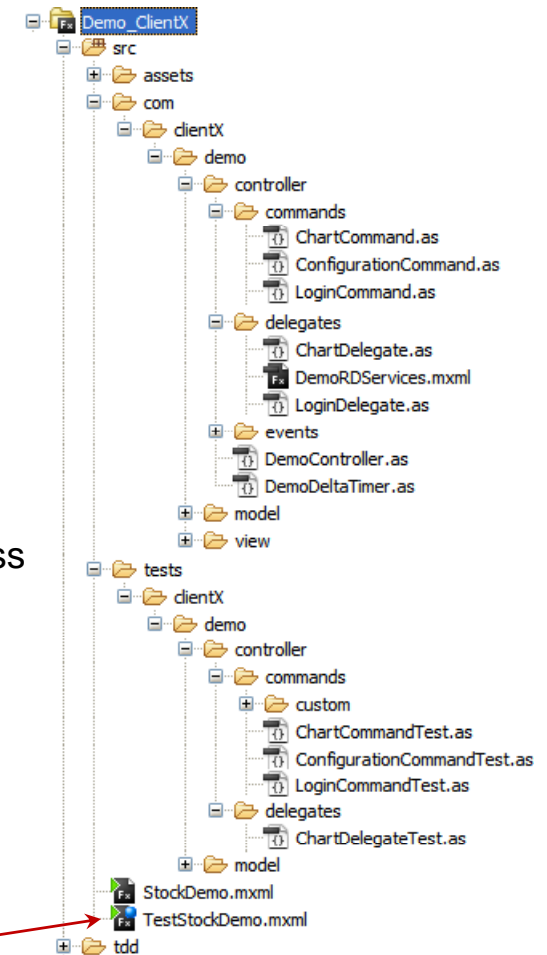
Code changes to business CFCs are not mapped to TestCases

... because the “other” developers are lazy or forget that detail.







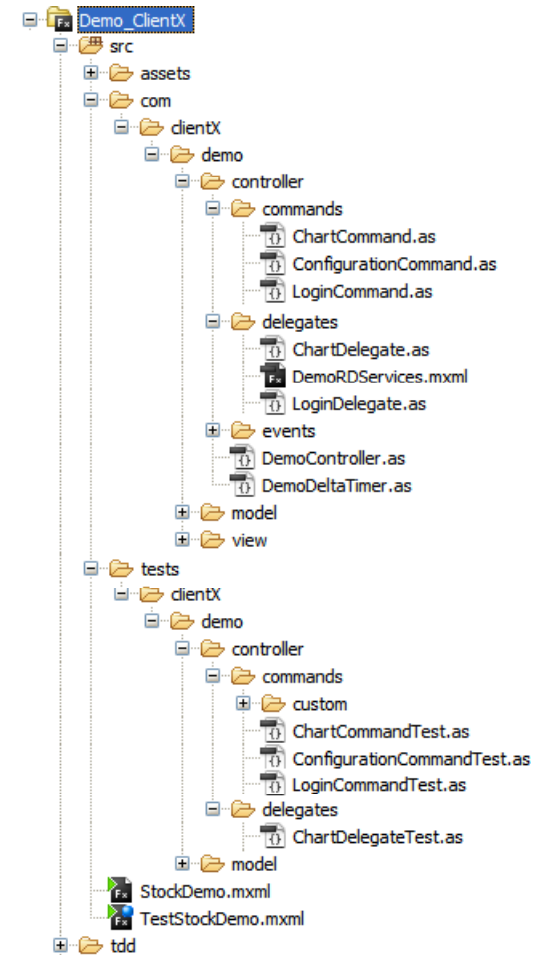
Updated the TestCases, but **forgot to update** the TestRunner

... the new #@#\$! tests never run.

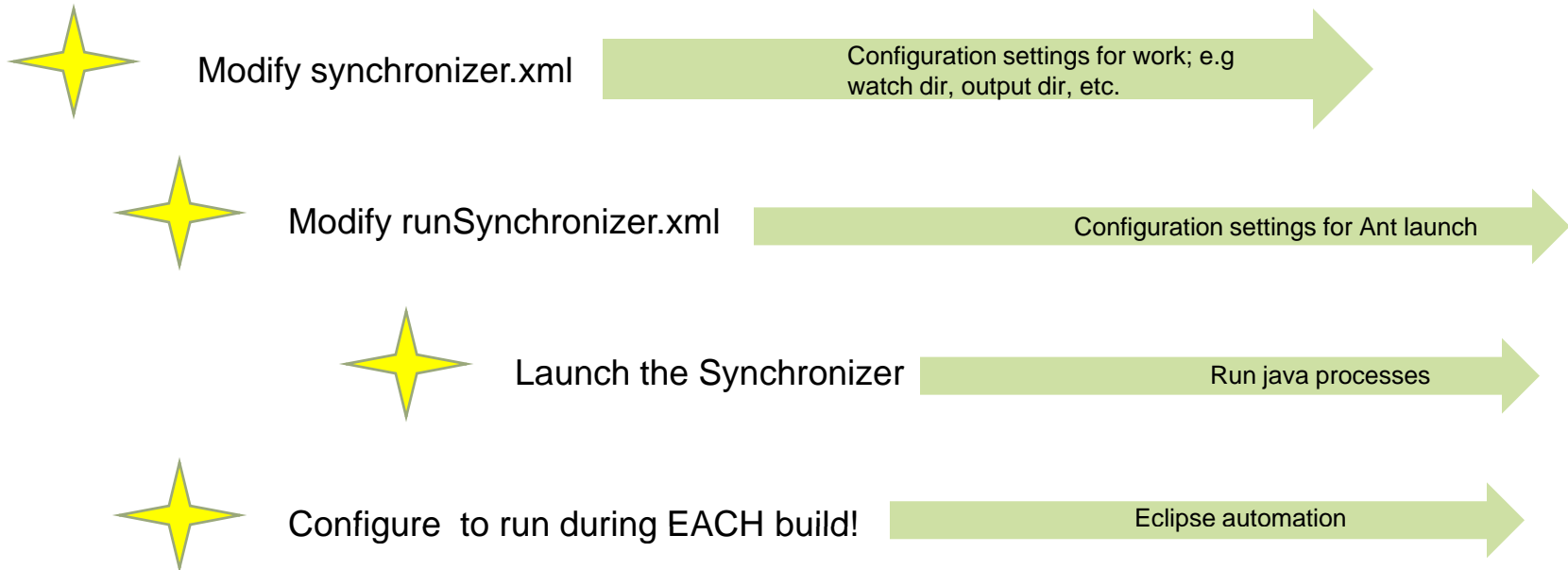


Introducing the **Synchronizer**

-  Always watches the business components
-  Always synchronizes TestCase class & methods
-  Does the CRUD “shells”
-  Auto-updates the TestRunner



Configuring the Synchronizer



Let the **Tools** do the work!



Use Test Framework



Use Cairngorm (UM)



Use ANT



Use Automated Builds



Test before commits to SVN

Resources for Continuous Testing

- **Testing Frameworks**

<http://code.google.com/p/as3flexunitlib/>

<http://www.darronschall.com/weblog/archives/000216.cfm/> [Getting Started w/ FlexUnit]

- **Dev Tools**

http://subversion.tigris.org/getting_subversion.html

<http://cruisecontrol.sourceforge.net/>

<http://ant.apache.org/>

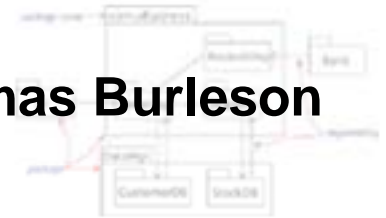
- **Auto-run ANT Tasks**

http://www.corfield.org/blog/index.cfm/do/blog.entry/entry/Automated_Testing_with_cfcUnit

- <http://www.fusionauthority.com/Techniques/4583-Test-Driven-Development-with-ColdFusion-Part-III.htm>



Thomas Burleson



**PRINCIPAL ARCHITECT
CERTIFIED INSTRUCTOR**
Adobe® Flex™



ThomasB@UniversalMind.com

<http://www.thomasburleson.biz/>