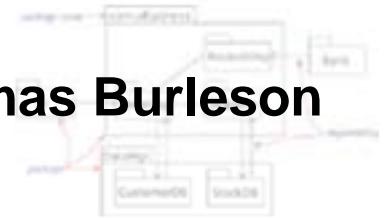




Thomas Burleson



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



ThomasB@UniversalMind.com

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

Continuous Testing

with

Coldfusion & Flex

This is **NOT**...



Not another session on “**Deep Dive** into using...”

We already have great online resources for the details of each framework: CFUnit, CFCUnit, Flex Unit



Not a session on “**How to configure** server builds...”

We already have great online resources for CFUnit, CFCUnit, Flex Unit



Where you will learn how to write the “**body**” of your test **logic!**

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

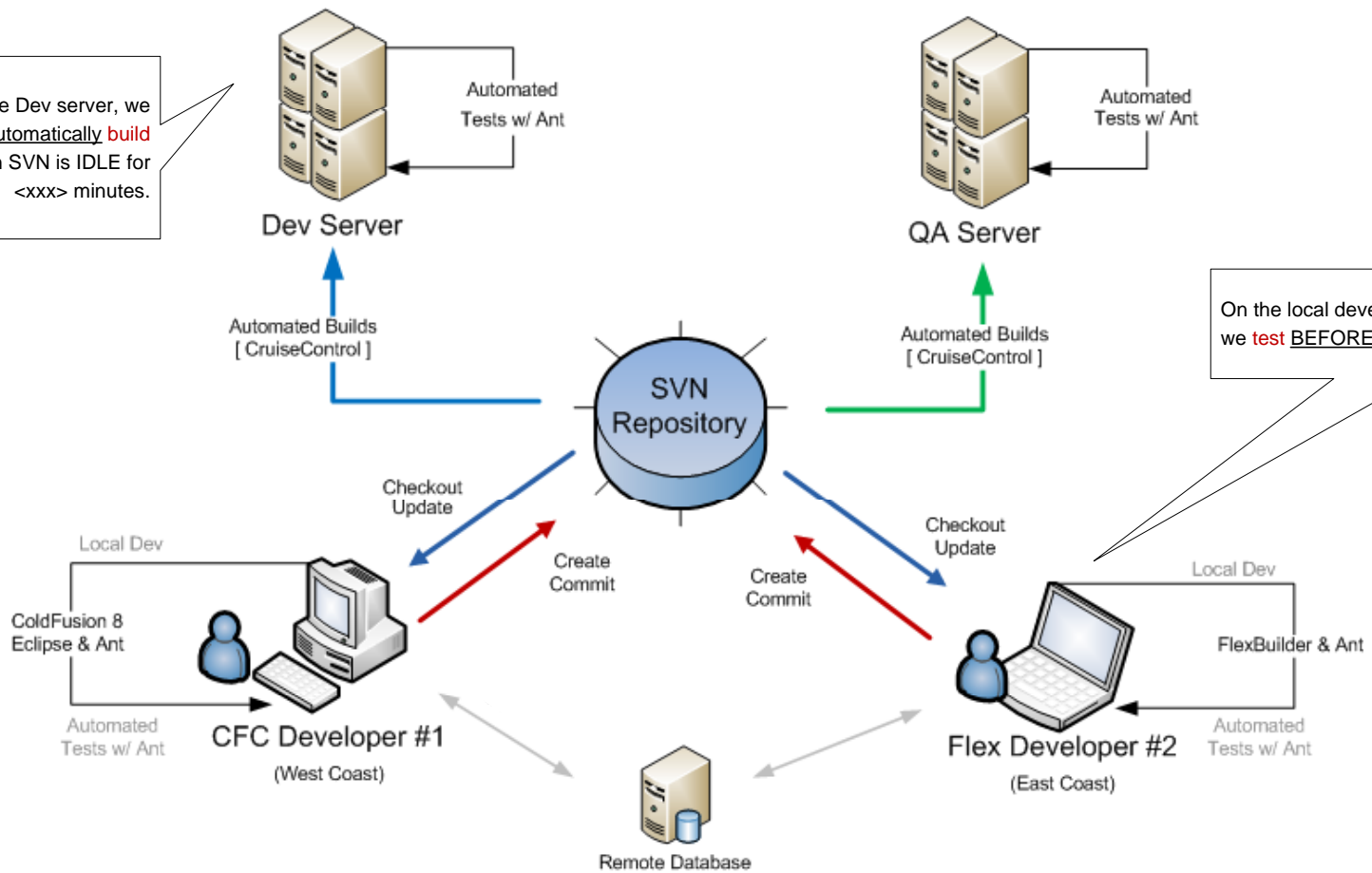


Continuous Integration:

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

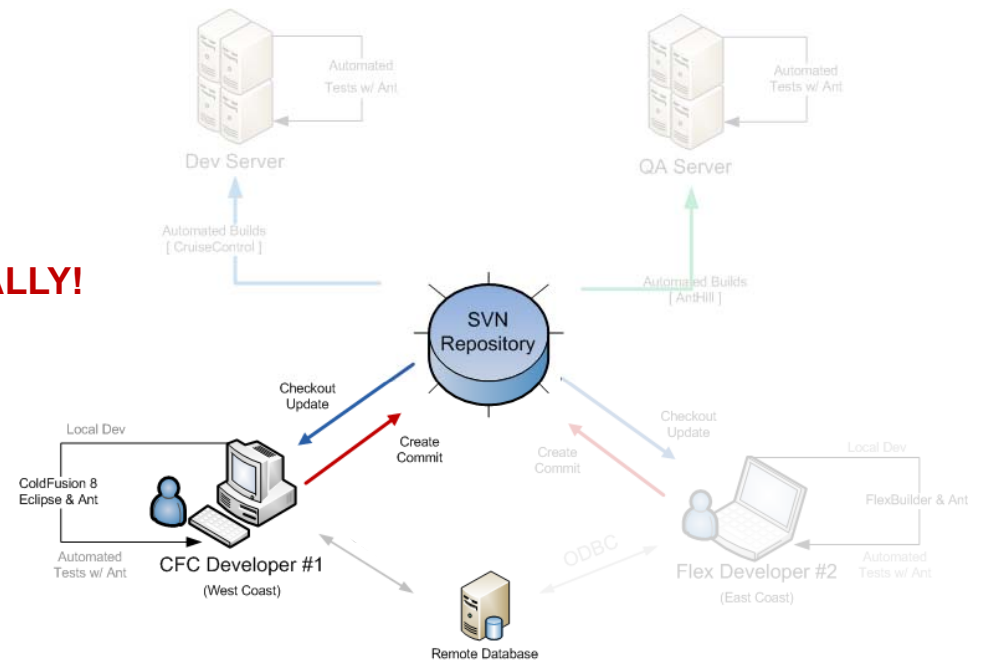
So **why** this session “Continuous Testing”?



Improve your development style

Improve quality of Unit Testing

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





“If you are not consistently testing, you are **crazy!”**

... no repeatable tests?
... no **regression** testing?



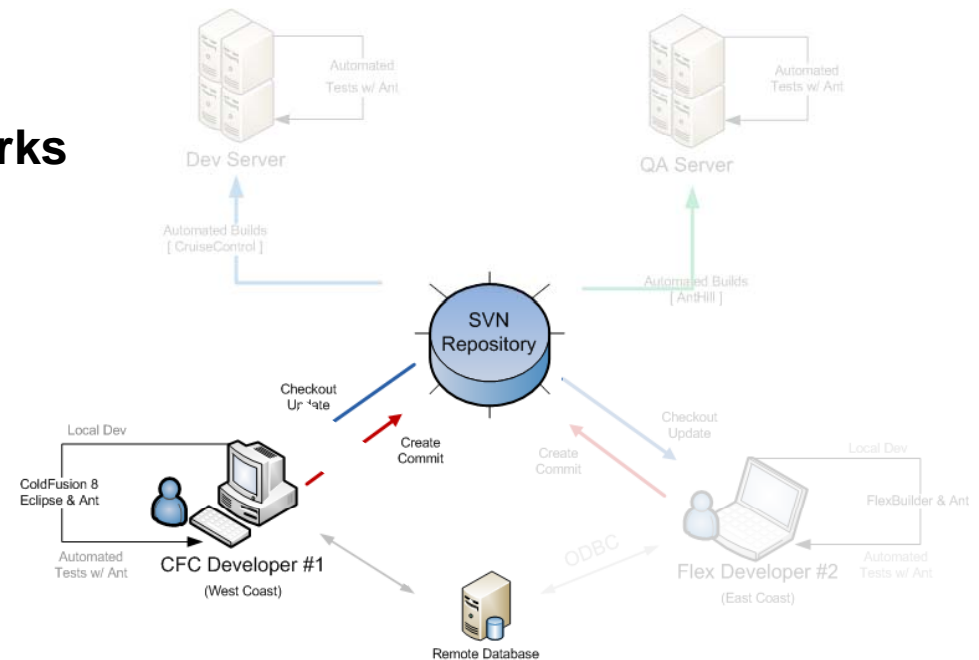
No more excuses!



Let the “tools” do the work.

Overview

- 1) Traditional Approach [CFML]
- 2) Using Testing Framework(s)
- 3) **Misconceptions**
- 4) **Problems** with Testing Frameworks
- 5) Introducing the “Synchronizer”
- 6) Let the “tools do the work...”
- 7) Resources....



Traditional Approach

Step 1:

Create 1 or more CFCs
[each with 1 or more methods]

Step 2:

Create 1 or more CFML pages to **test** & **debug** the CFC calls



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



Issues with Traditional Approach

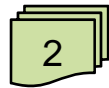
It is Ad-HOC (makeshift...)!



How do make sure you test each method of an CFC?
... and no way to easily test all .cfcs



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

Testing Frameworks

CFUnit - [ColdFusion](#)

CFCUnit - [ColdFusion](#)

FlexUnit - [Flex](#)

ASUnit - [Flash](#)

Step 1:

Quick Review of CFUnit

a) Installation

b) Creating / Using TestCases

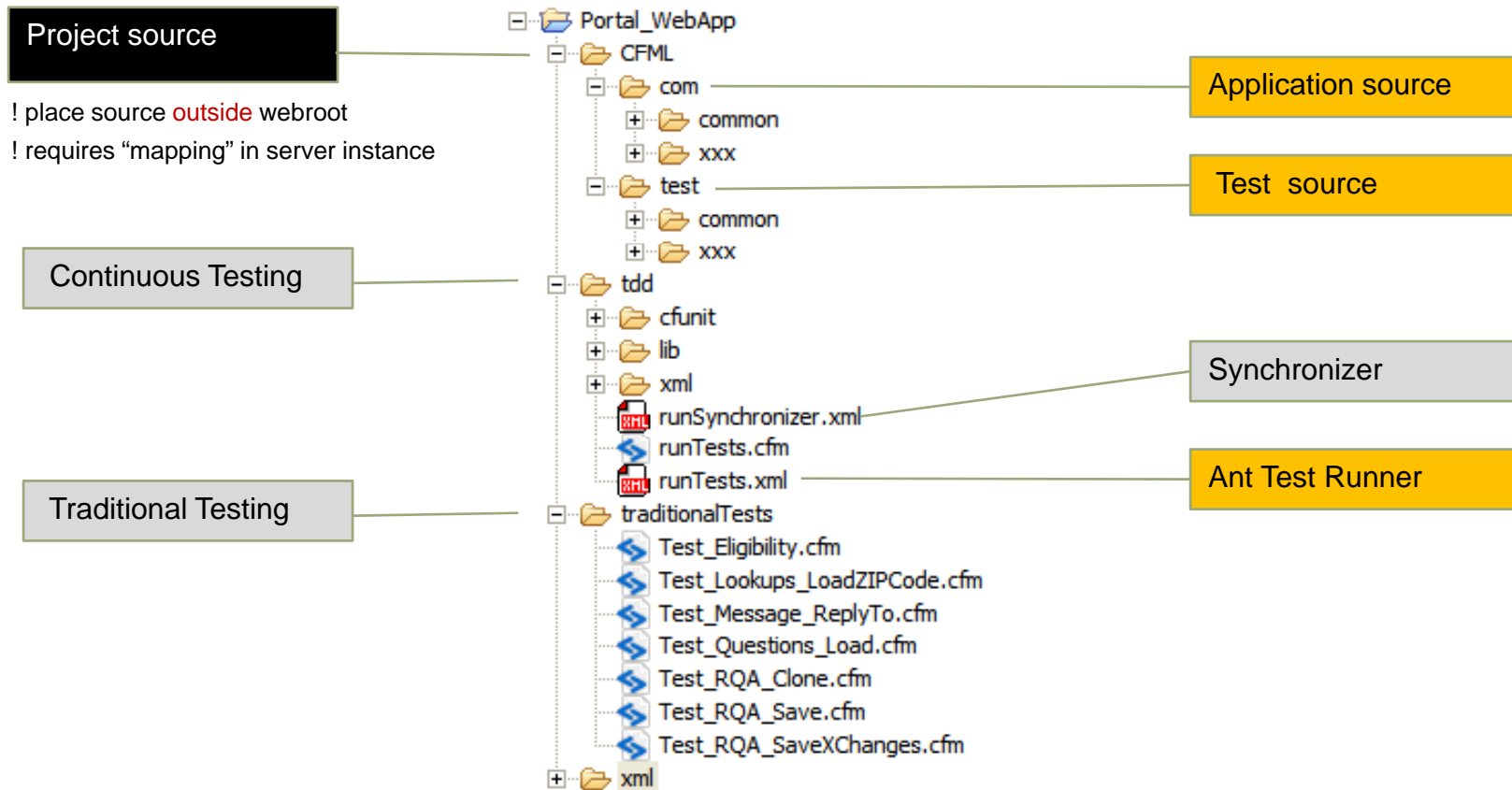
Step 2:

Using TestRunner to run the TestCases

Step 3:

Using Eclipse Ant to run the TestCases

Sample Project



Test Mappings

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

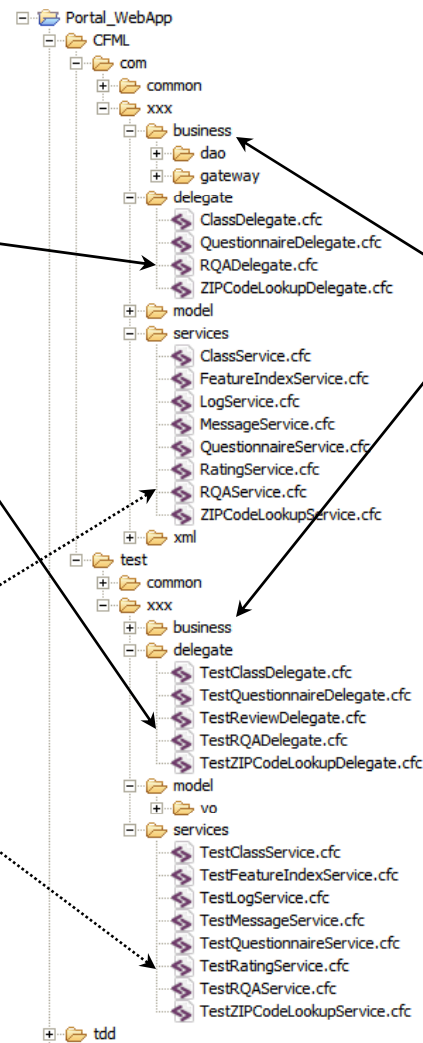
ZipCodeLookup.cfc
TestZipCodeLookup.cfc

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

<cffunction name="getDetailsByCode"
<cffunction name="testGetDetailsByCode"

- ❖ 1-to-1 mappings of package names

com.xxx.business.dao
test.xxx.business.dao



Running TestRunner

Use CFML:

<http://localhost:9600/portalApp/tdd/runTests.cfm>

Use Ant:

[Run as Ant Build <webroot>/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

```

Using **FlexUnit**



Using FlexUnit Framework



Building FlexUnit Testcases



Configure TestRunner w/ TestCases



Launch TestRunner

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

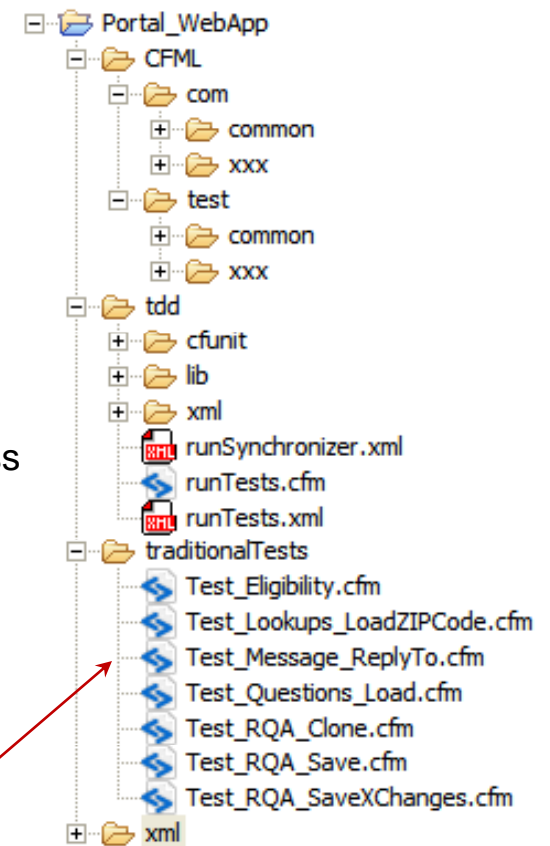
Step 1: write the test to verify the bug exists

Step 2: fix the bug





Step 3: run the test to verify

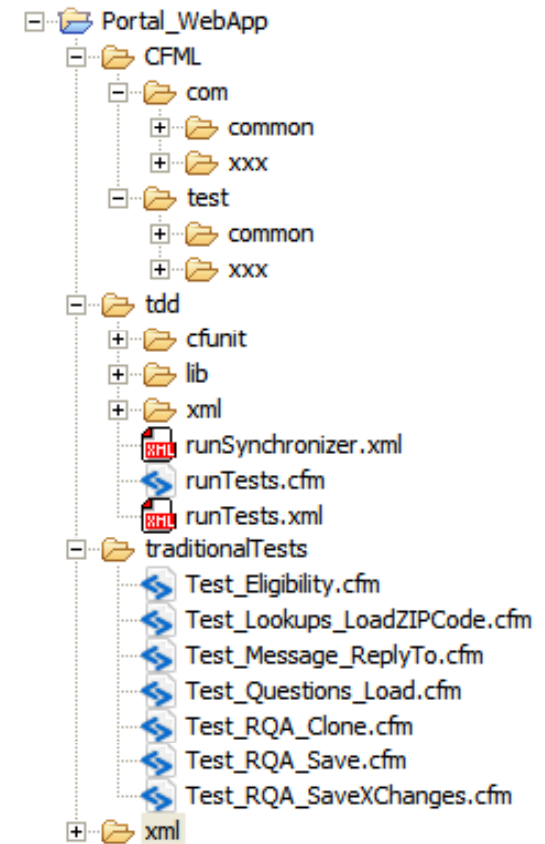
Why are people **resistant** to testing?

-  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.

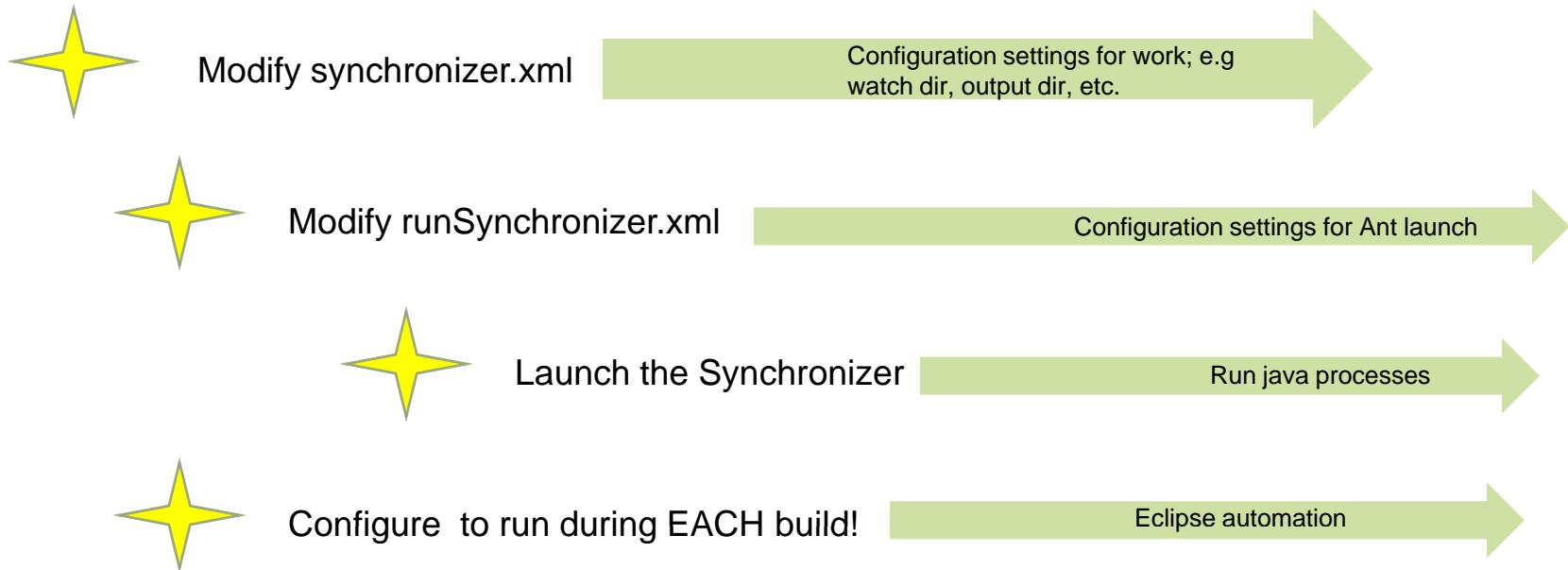


Introducing the **Synchronizer**

-  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 Eclipse Ant



Use Synchronizer



Use SVN



Use Automated Builds

Resources for Continuous Testing

- **Testing Frameworks**

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

- <http://sourceforge.net/projects/cfunit/>

- <http://www.cfcunit.org/cfcunit/>

- **Dev Tools**

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

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

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

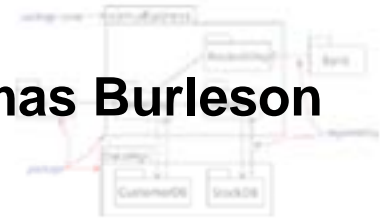
- **Auto-run ANT Tasks**

- 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/>