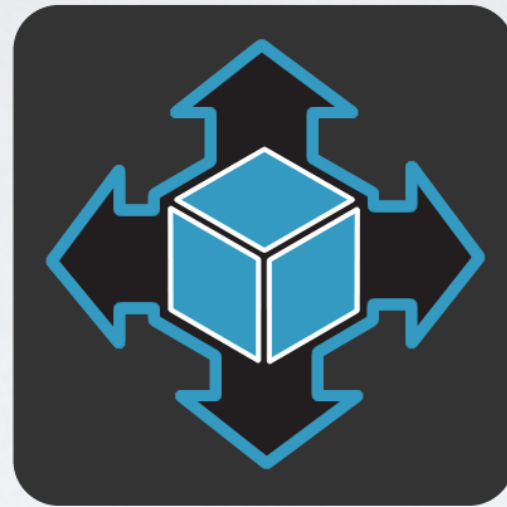


xUnit and BDD Testing Goodness!



TESTBOX

TESTBOX

WHO AM I?

WHO AM I?

- Luis Majano - Computer Engineer
- Born in El Salvador ----->
- Architecture + Software Design
- CEO of Ortus Solutions
- Manager of the IECFUG (www.iecfug.com)
- Adobe Community Professional
- Creator of all things Box:
ColdBox, ContentBox, WireBox....



COMMUNITY
PROFESSIONAL

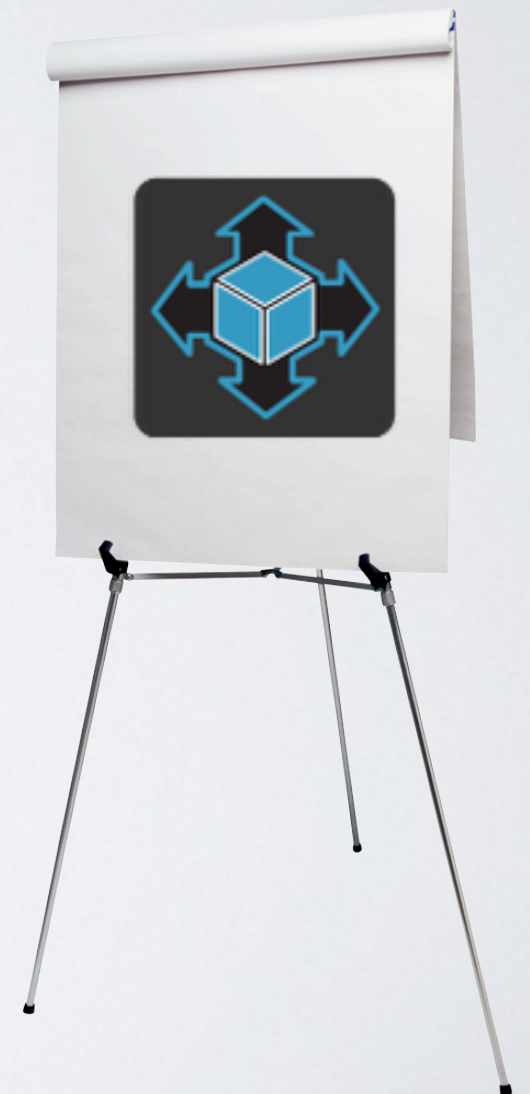


AGENDA



AGENDA

- Why TestBox
- Capabilities
- Installation
- MXUnit
- Testing Styles
- Deep Dive







Why

**Testing
Landscape**



Why

Where to start?
What to test?
What not to test?

**Testing
Landscape**



TestBox is a next generation testing framework for ColdFusion that is based on BDD (Behavior Driven Development) for providing a clean obvious syntax for writing tests. It contains not only a testing framework, runner, assertions and expectations library but also integrates with MockBox for mocking and stubbing. It also supports xUnit style of testing and MXUnit compatibilities.

SET
GOAL

A staircase diagram is drawn in white chalk on a blackboard. It consists of five steps, each represented by a horizontal line segment followed by a vertical line segment going up. The steps are arranged in a diagonal line from the bottom-left to the top-right. Each step has a colored sticky note attached to it, with the following text from bottom-left to top-right: 'SET GOAL' (light blue), 'MAKE PLAN' (light green), 'GET TO WORK' (yellow), 'STICK TO IT' (orange), and 'REACH GOAL' (pink).

MAKE
PLAN

GET
TO
WORK

STICK
TO IT

REACH
GOAL

- BDD & xUnit style testing



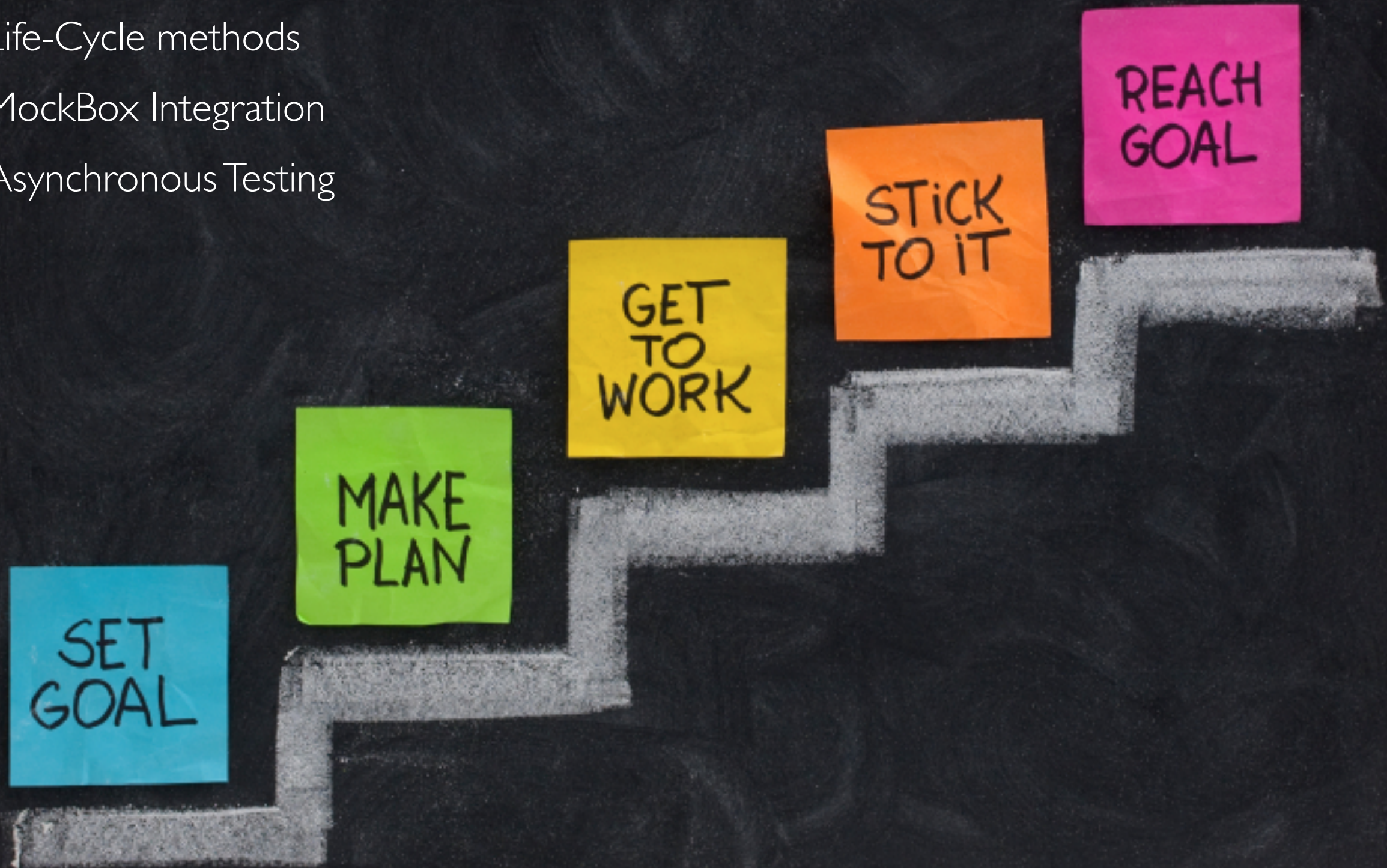
- BDD & xUnit style testing
- Life-Cycle methods



- BDD & xUnit style testing
- Life-Cycle methods
- MockBox Integration



- BDD & xUnit style testing
- Life-Cycle methods
- MockBox Integration
- Asynchronous Testing



- BDD & xUnit style testing
- Life-Cycle methods
- MockBox Integration
- Asynchronous Testing



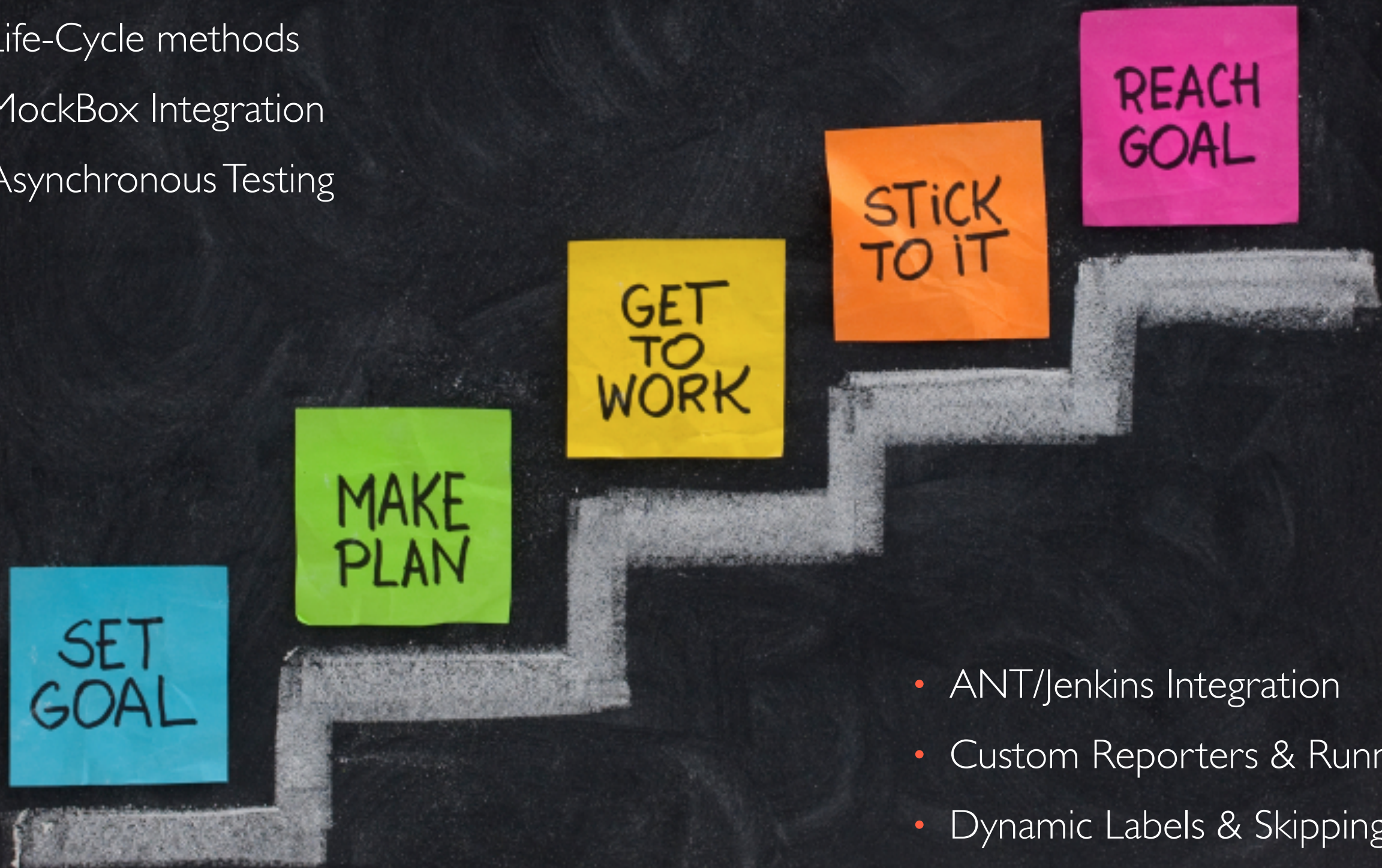
- ANT/Jenkins Integration

- BDD & xUnit style testing
- Life-Cycle methods
- MockBox Integration
- Asynchronous Testing



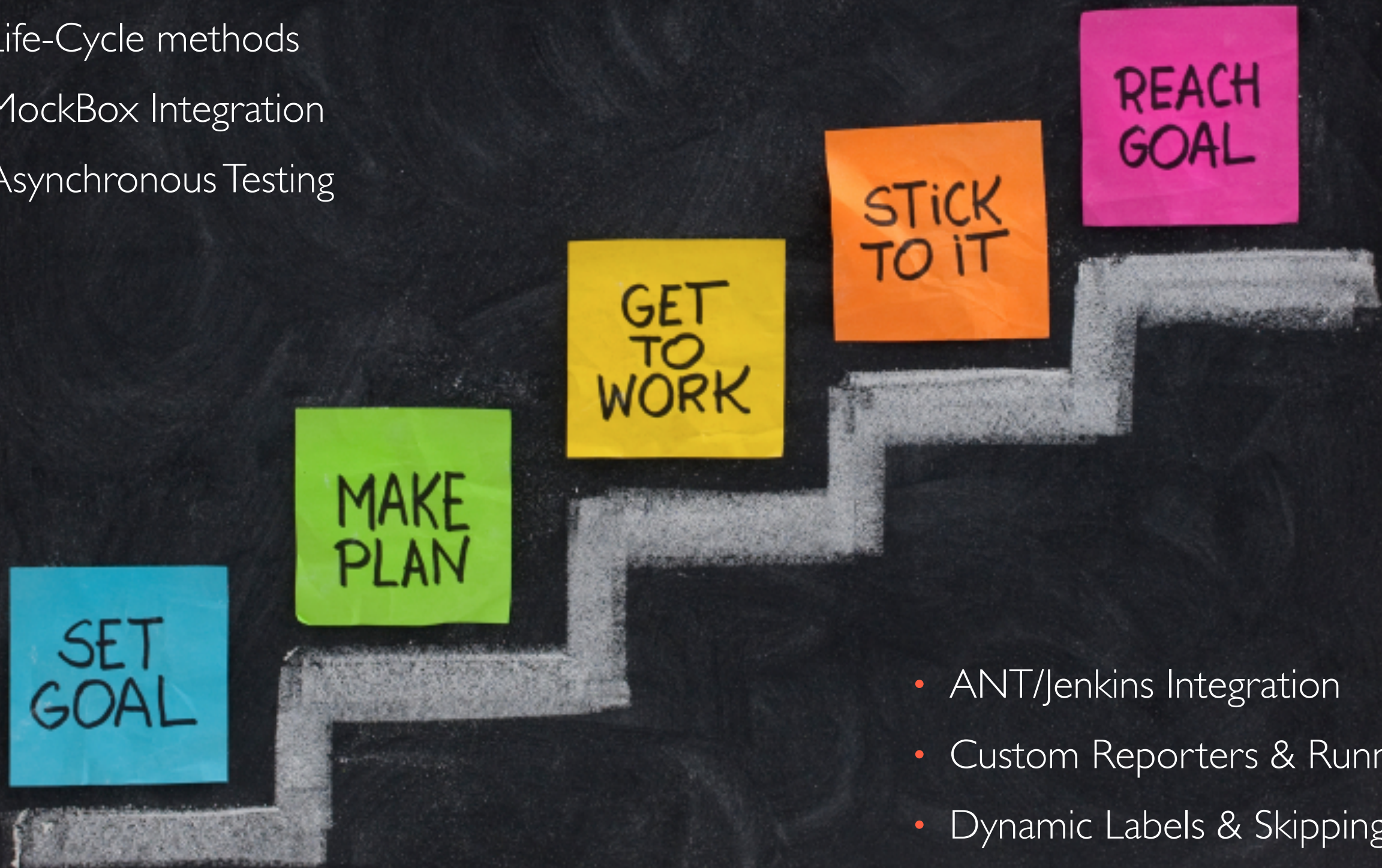
- ANT/Jenkins Integration
- Custom Reporters & Runners

- BDD & xUnit style testing
- Life-Cycle methods
- MockBox Integration
- Asynchronous Testing



- ANT/Jenkins Integration
- Custom Reporters & Runners
- Dynamic Labels & Skipping

- BDD & xUnit style testing
- Life-Cycle methods
- MockBox Integration
- Asynchronous Testing



- ANT/Jenkins Integration
- Custom Reporters & Runners
- Dynamic Labels & Skipping
- Debug Output Streams

INSTALLATION + REQUIREMENTS

- ColdFusion 9.0.1+, Railo 3.1+
 - xUnit + MXUnit Compatibility
- ColdFusion 10+, Railo 4+
 - xUnit, MXUnit, BDD
- Place anywhere you like, create a “/testbox” mapping

```
<cfset this.mappings[ "/testbox" ] "/my/path/testbox">
```

- Using ColdBox? Already installed!
- Sublime Package, CPU is coming
















MXUNIT COMPATIBLE

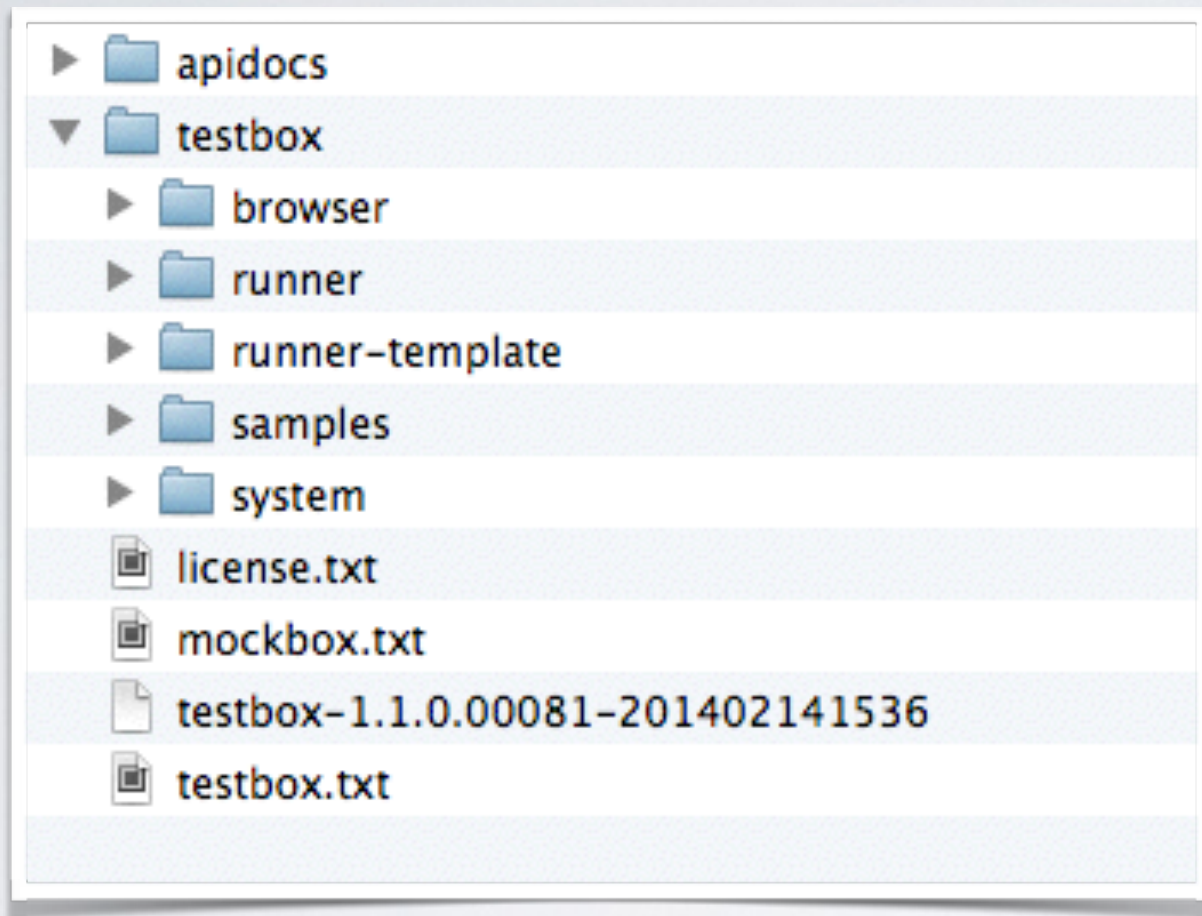
- Compatible with xUnit style by MXUnit
- Migrate existing tests to TestBox
- No BDD
- How do you migrate?

```
<cfset this.mappings[ "/mxunit" ] expandPath( "/testbox/system/testing/compat" )>  
<cfcomponent extends="mxunit.framework.TestCase">  
<cfcomponent extends="testbox.system.testing.compat.framework.TestCase">
```

- If something is not working, report it: bugs@coldbox.org

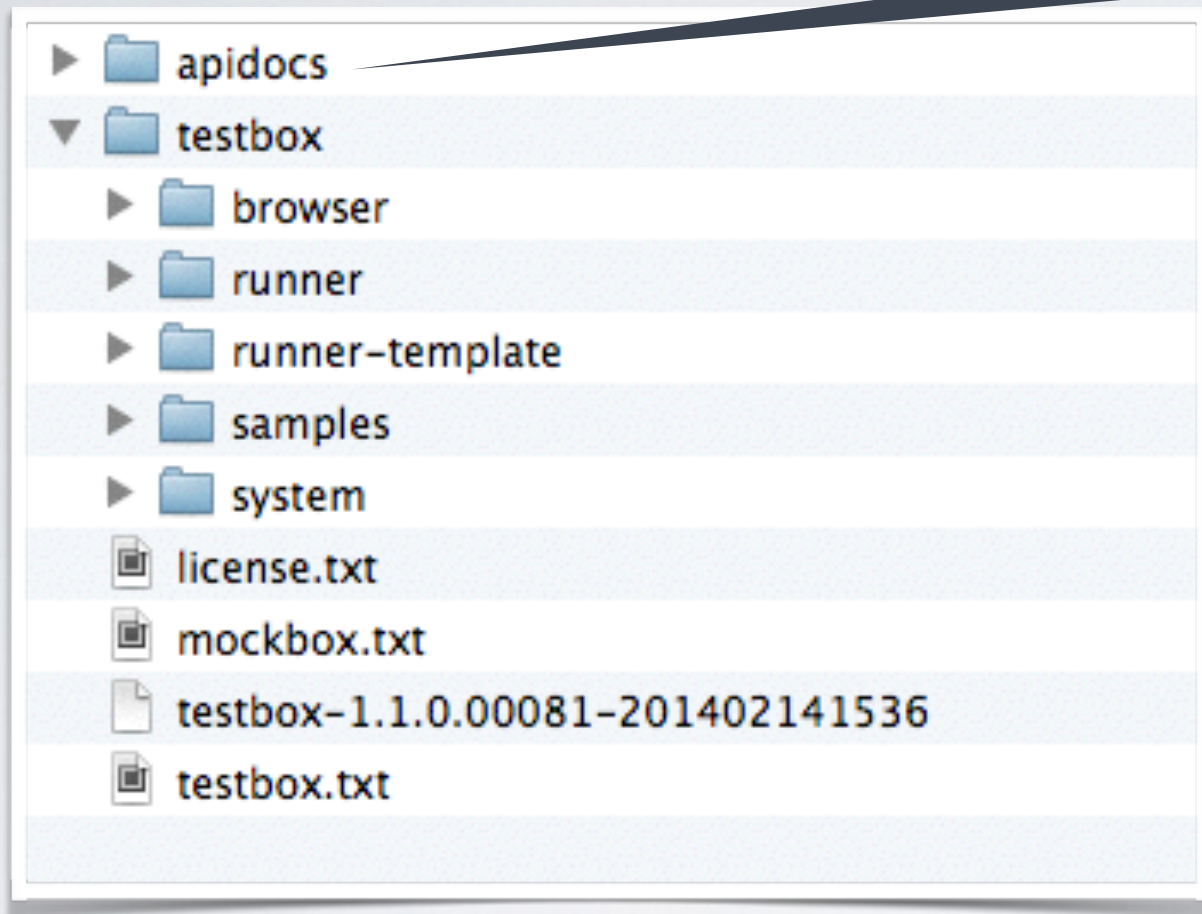
- ▶  apidocs
- ▼  testbox
 - ▶  browser
 - ▶  runner
 - ▶  runner-template
 - ▶  samples
 - ▶  system
-  license.txt
-  mockbox.txt
-  testbox-1.1.0.00081-201402141536
-  testbox.txt

What you get!



What you get!

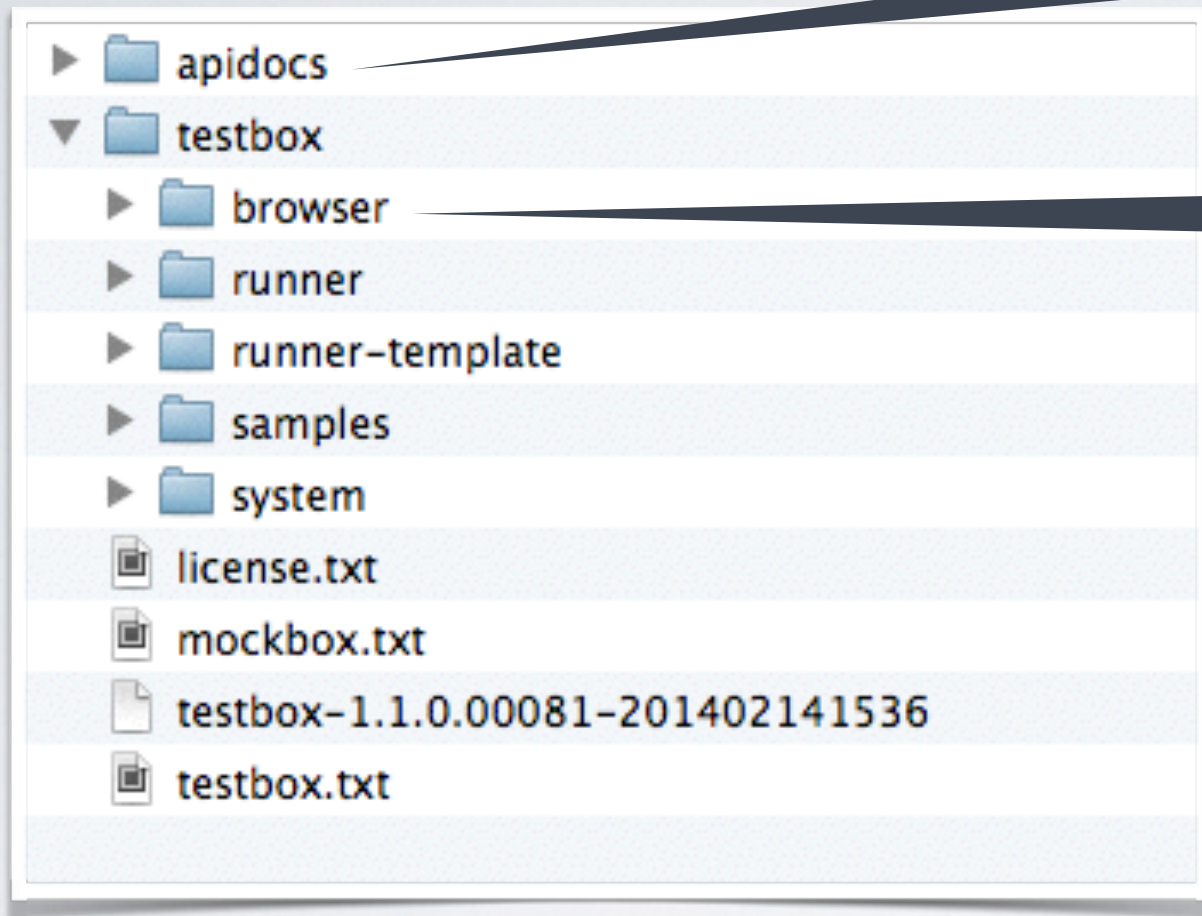
API Docs



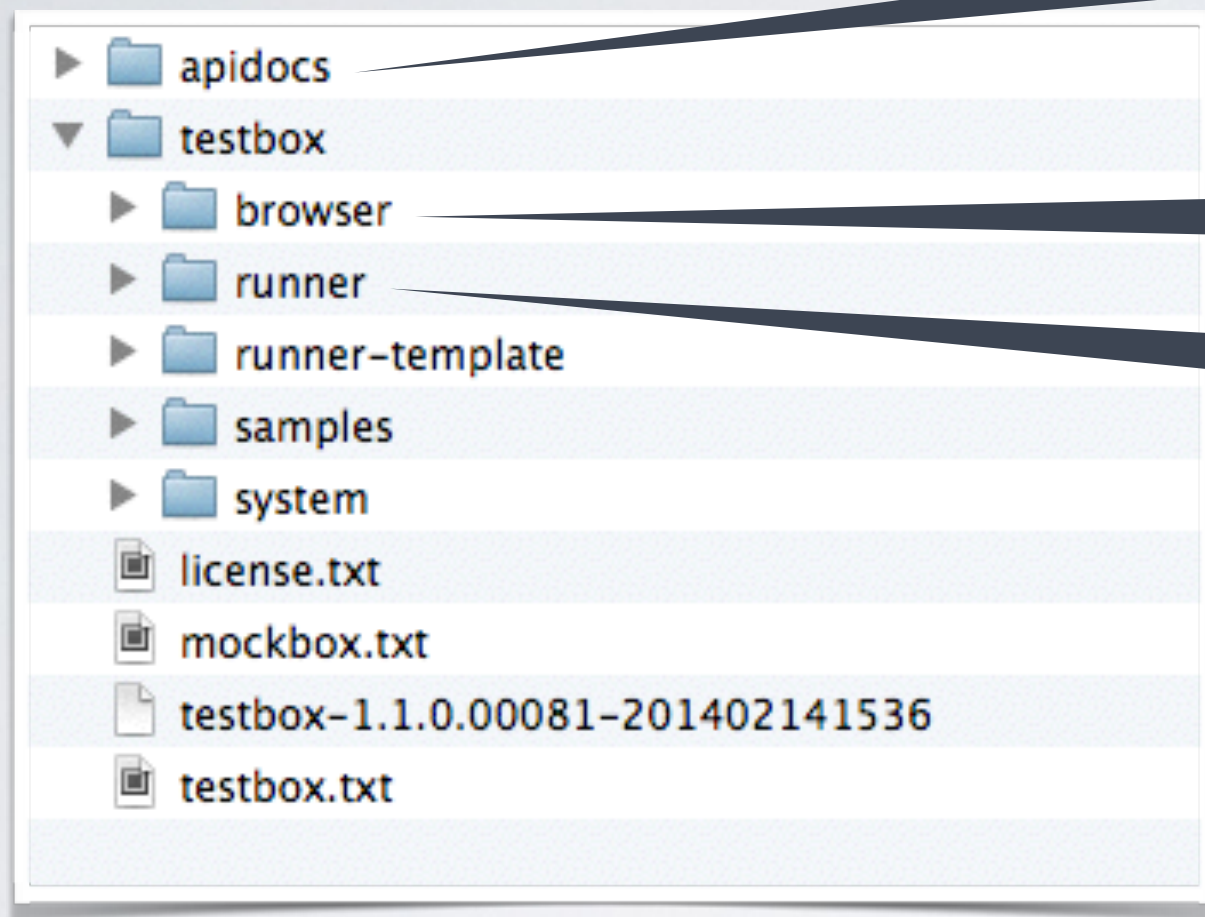
What you get!

API Docs

Test Browser



What you get!

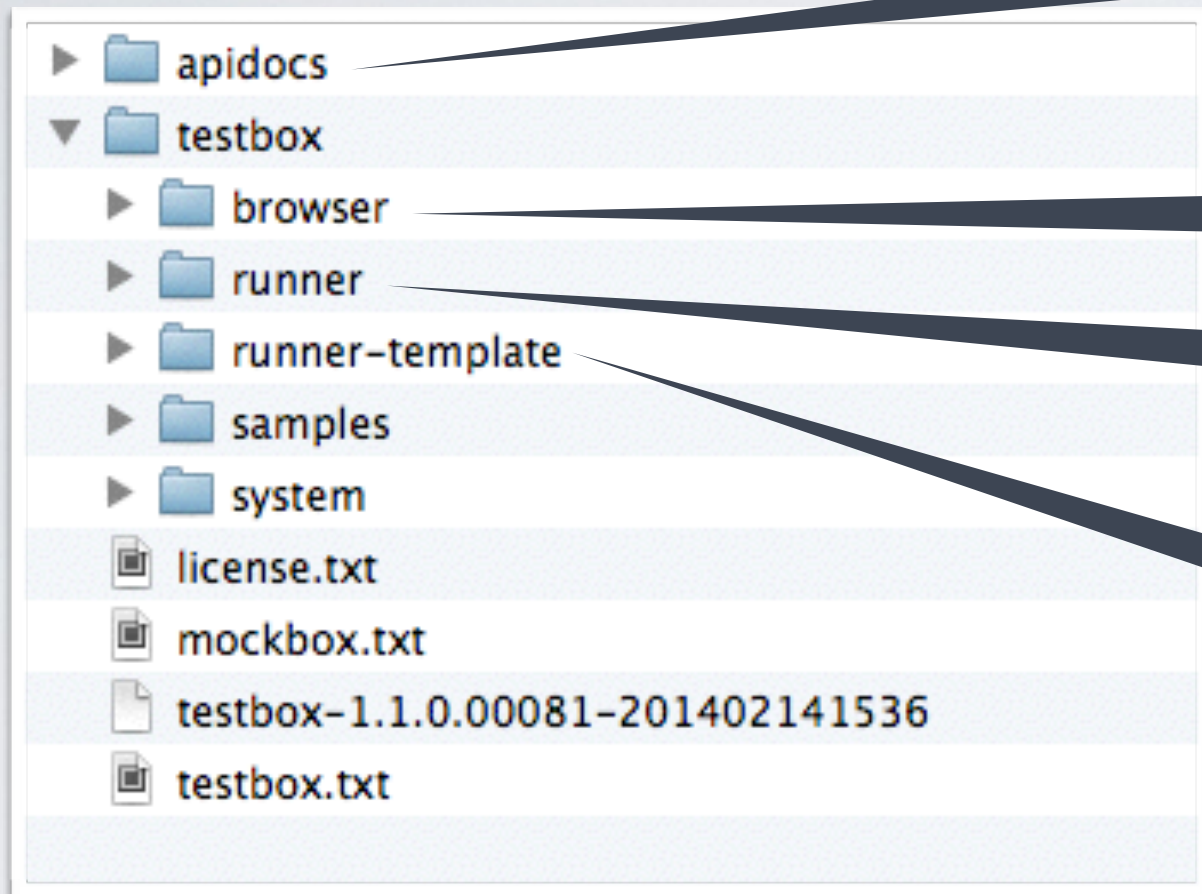


API Docs

Test Browser

Global Runner

What you get!



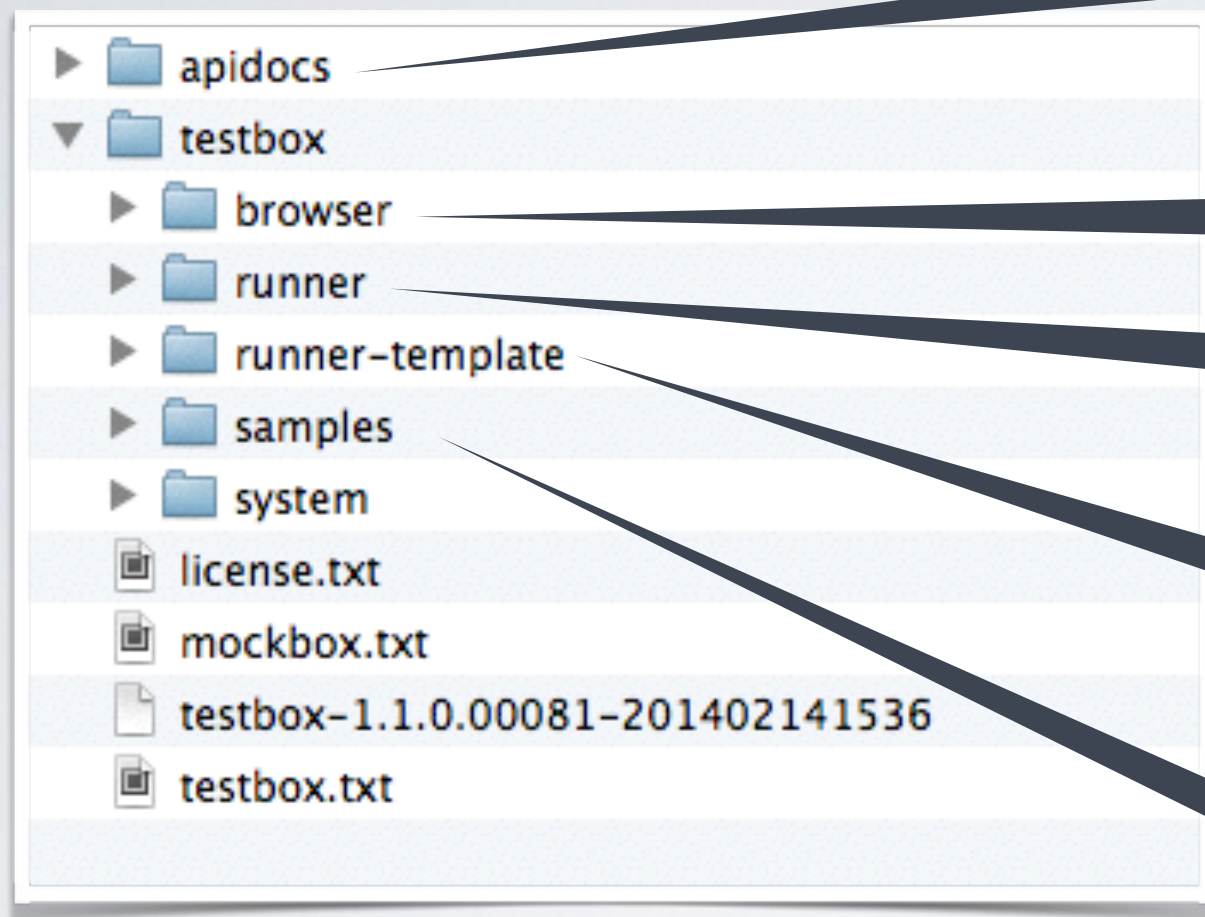
API Docs

Test Browser

Global Runner

Test Harness

What you get!



API Docs

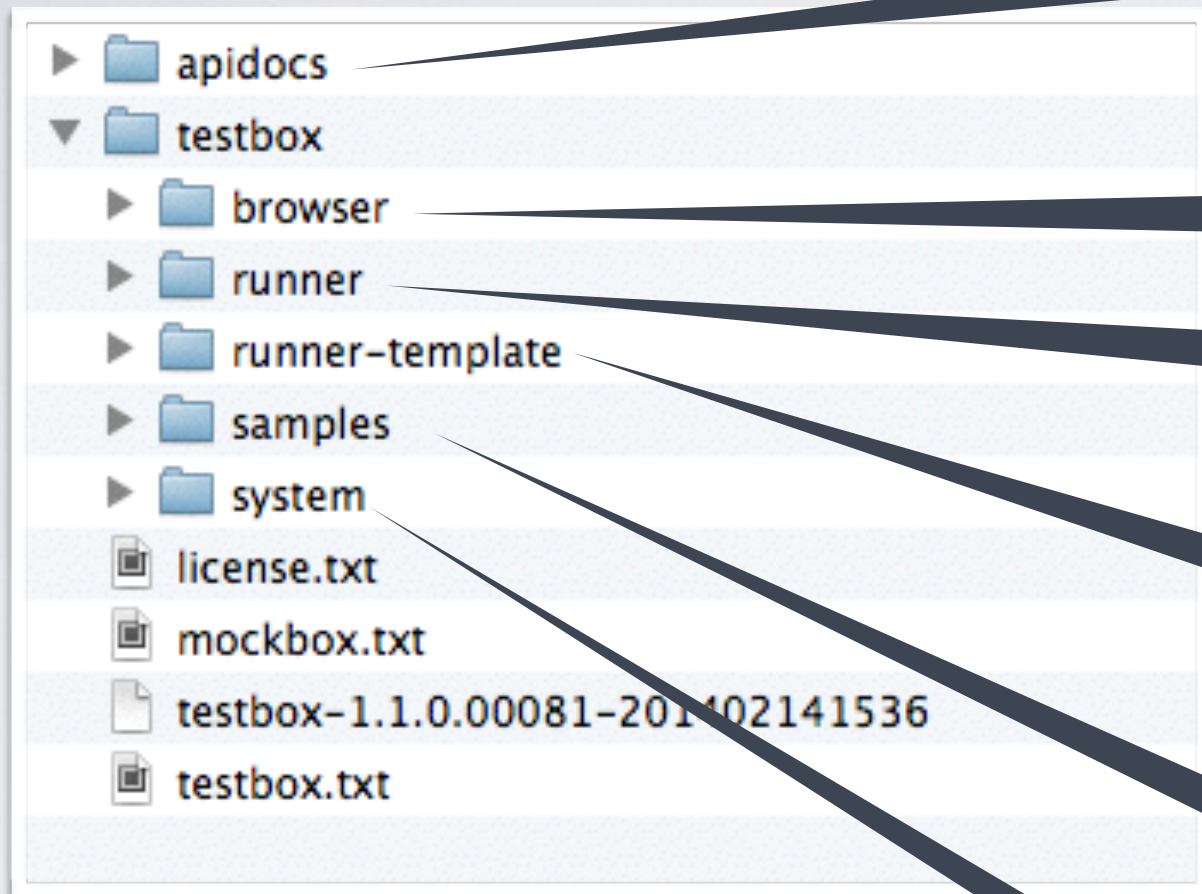
Test Browser

Global Runner

Test Harness

Samples

What you get!



API Docs

Test Browser

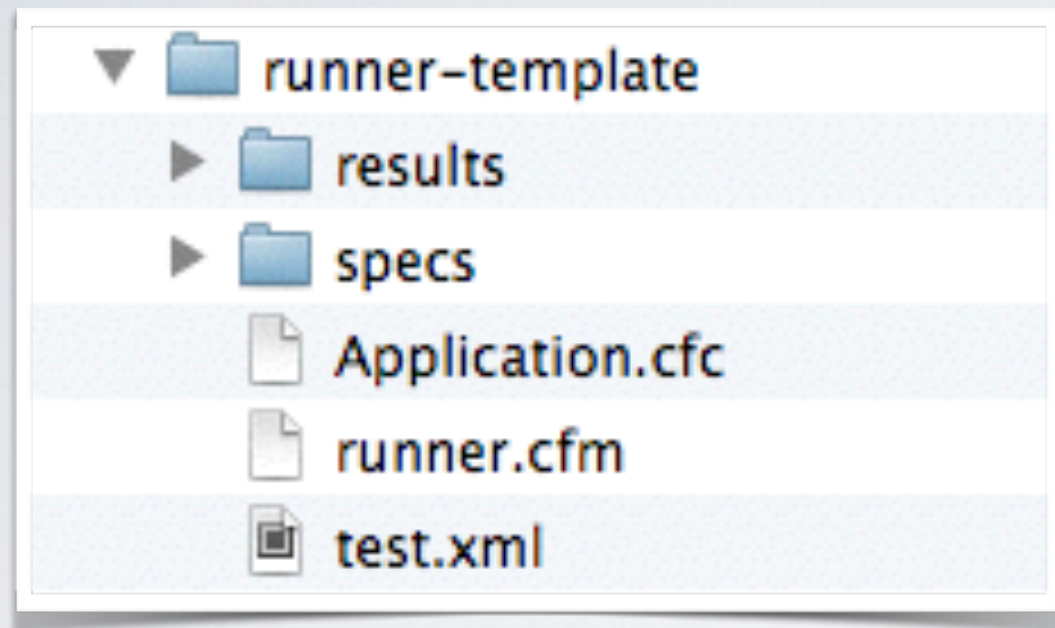
Global Runner

Test Harness

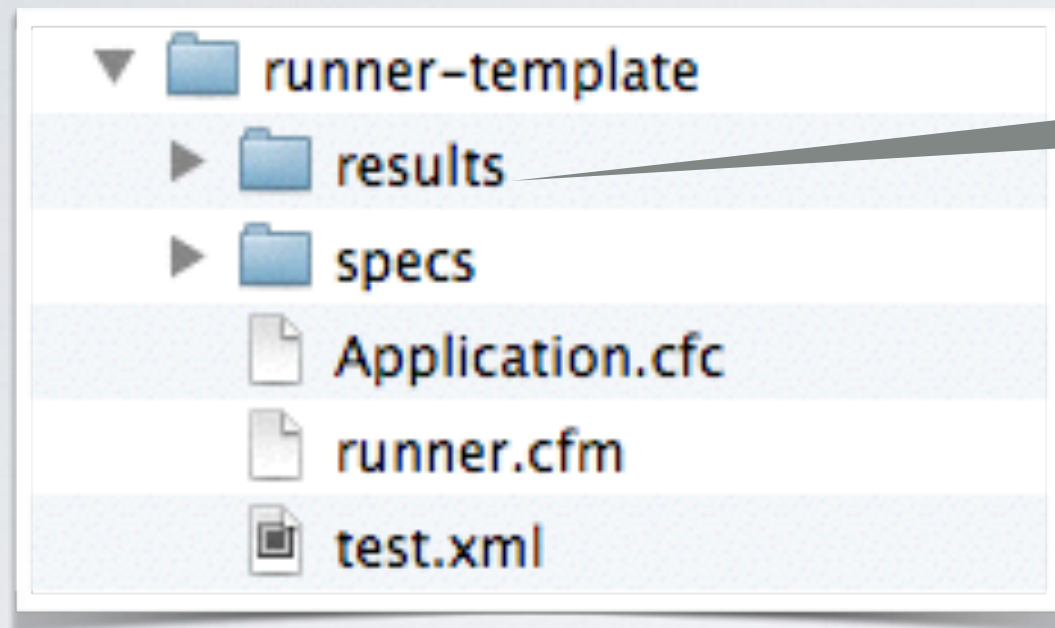
Samples

Core

Test Harness

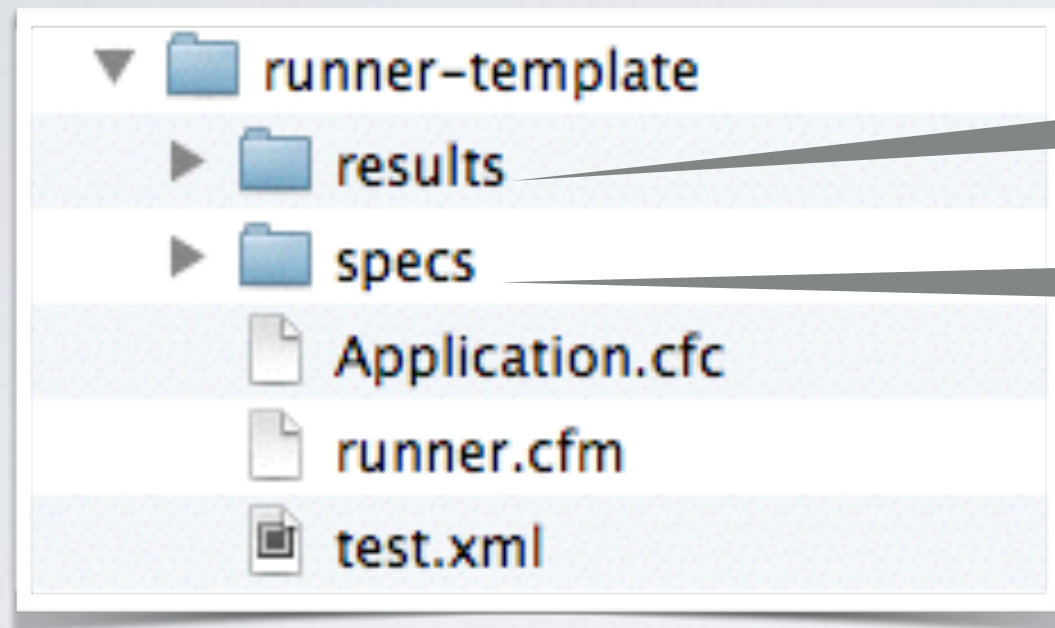


Test Harness



Automated test results!

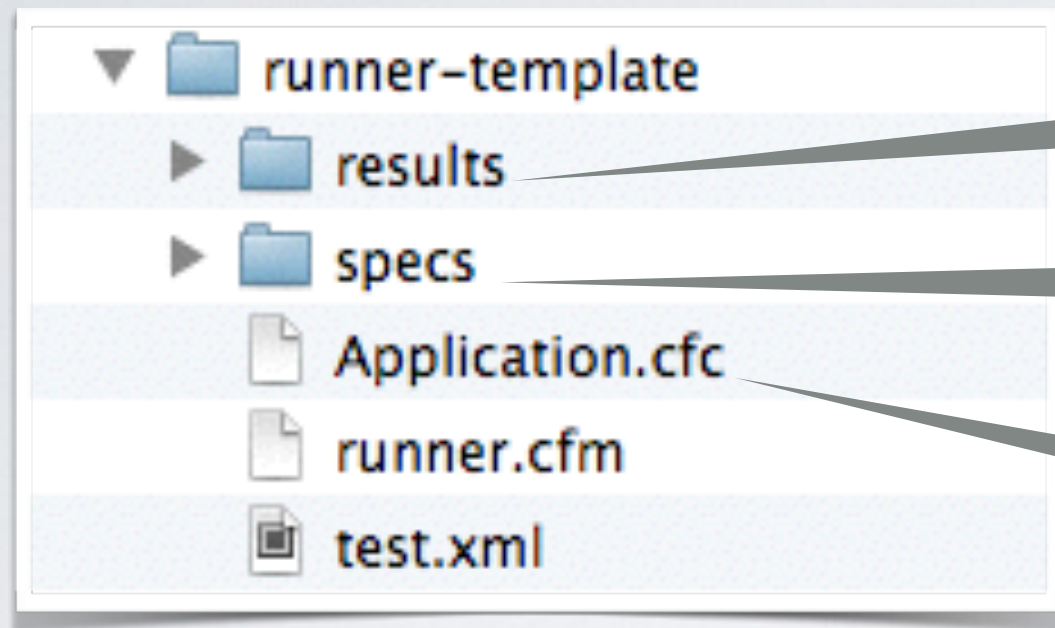
Test Harness



Automated test results!

xUnit/BDD Test Bundles

Test Harness

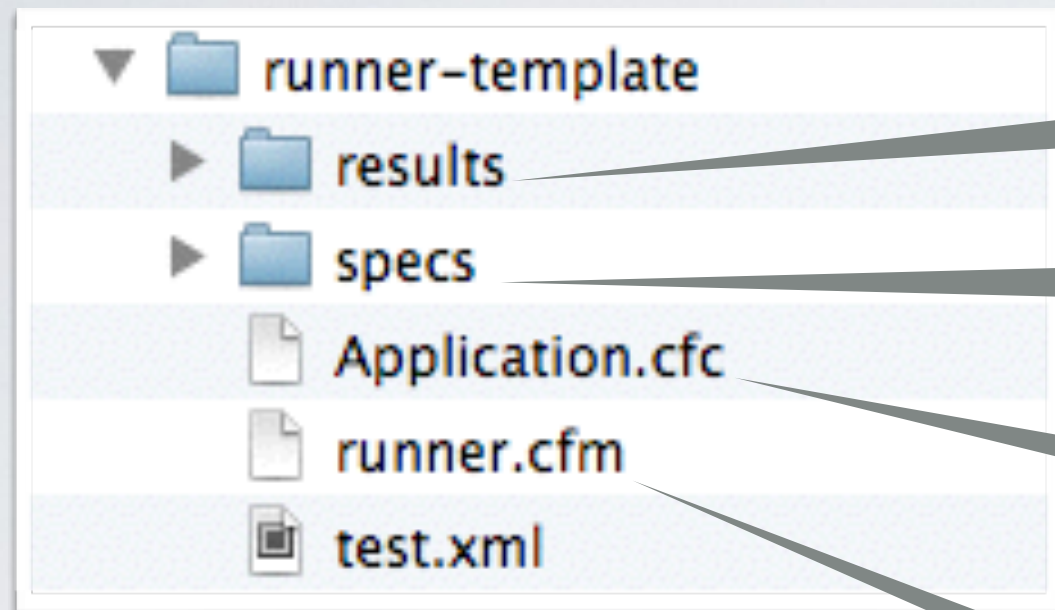


Automated test results!

xUnit/BDD Test Bundles

Harness bootstrap

Test Harness



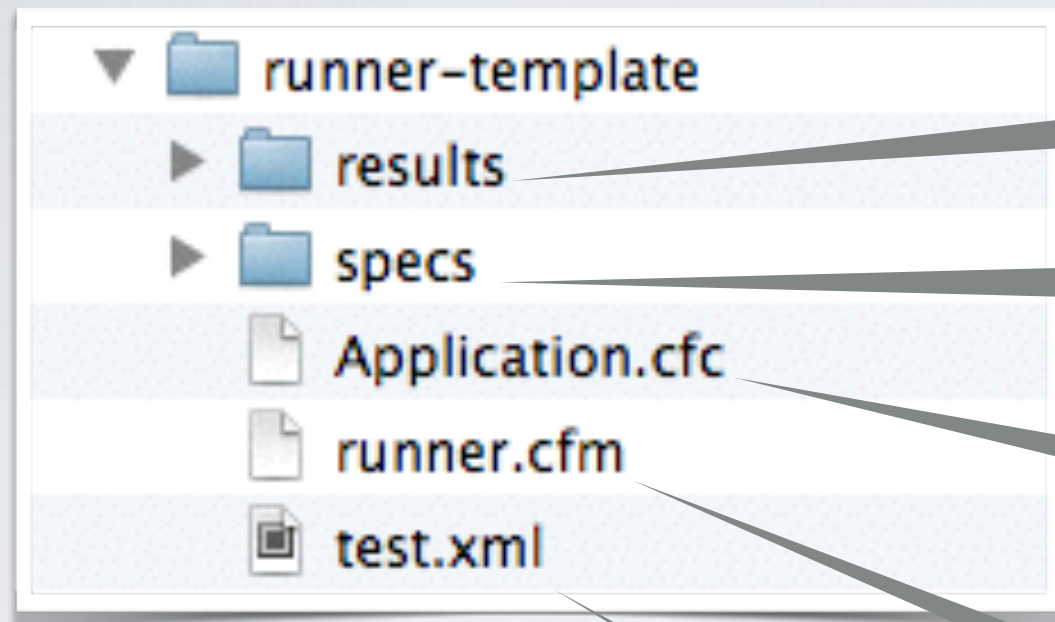
Automated test results!

xUnit/BDD Test Bundles

Harness bootstrap

HTML Runner

Test Harness



Automated test results!

xUnit/BDD Test Bundles

Harness bootstrap

HTML Runner

ANT Runner



TESTING STYLES



TESTING STYLES

xUnit

TDD

Unit Focused

Function Focused

Asserts

Asserts

BDD

Test Scenarios

Spec Focused

Nested Scenarios

Expectations

Expectations



TEST BUNDLE CFC

- No matter what style, you start with a test bundle CFC
- Inherits from **testbox.system.testing.BaseSpec** or **not!**

```
component extends="testbox.system.testing.BaseSpec"{}  
component {}
```

- URL runner caveat
- Get's lots of methods and properties for testing
- TesBox will then execute all tests within 1 or more bundles



RUNNING YOUR BUNDLES

- Execute bundle (if using inheritance) via the URL
 - **`http://mysite/test/bundle.cfc?method=runRemote`**
- Using the TestBox Class: **`testbox.system.testing.TestBox`**
 - Bundle(s) Runner
 - Directory Runner
 - SOAP Runner
 - HTTP/REST Runner
 - ANT Runner
 - Custom Runners
- What's the output? We call this reporters



- **ANTJunit** : A specific variant of JUnit XML that works with the ANT junitreport task
- **Codexwiki** : Produces MediaWiki syntax for usage in Codex Wiki
- **Console** : Sends report to console
- **Doc** : Builds semantic HTML to produce nice documentation
- **Dot** : Builds an awesome dot report
- **JSON** : Builds a report into JSON
- **JUnit** : Builds a JUnit compliant report
- **Raw** : Returns the raw structure representation of the testing results
- **Simple** : A basic HTML reporter
- **Text** : Back to the 80's with an awesome text report
- **XML** : Builds yet another XML testing report
- **Tap** : A test anything protocol reporter
- **Min** : A minimalistic view of your test reports



GLOBAL RUNNER



GLOBAL RUNNER



TESTBOX
v1.0.0.@build.number@

TestBox Global Runner

Please use the form below to run test bundle(s), directories and more.

☒ Recurse Directories

Simple Reporter ▾

Clear

Run

v1.0.0.@build.number@

TESTBOX

Simple Reporter ▾

Clear


Run



TEST BROWSER



TEST BROWSER


TESTBOX
v1.0.0.@build.number@
[Run All](#)

TestBox Test Browser:
Below is a listing of the files and folders starting from your root
`/Users/lmajano/Sites/cboxdev/core/coldbox/testing`. You can click on individual tests in order to execute them or click on the **Run All** button on your left and it will execute a directory runner from the visible folder.
Contents: `/coldbox/testing/cases/`
[<< Back](#)

- +aop
- +cache
- +core
 - EventHandlerTest.cfc
- index.cfm
- +interceptors
 - InterceptorTest.cfc
- +ioc
- +logging
- +mvc
- +orm
- +plugins
 - PluginTest.cfc
- +remote
- +testing
- +validation
- +web

- +mcp
- +validation
- +testing
- +remote
- +mcp



RUNNER SAMPLES



RUNNER SAMPLES

```
1 <cfsetting showdebugoutput="false" >
2 <cfscript>
3 r = new coldbox.system.testing.TestBox( directory={
4     mapping = "coldbox.testing.cases.testing.specs",
5     recurse = true,
6     filter = function( path ){ return true; }
7 });
8 </cfscript>
9 <cfoutput>#r.run(reporter="simple")#</cfoutput>
```




RUNNER SAMPLES

```
1 <cfsetting showdebugoutput="false" >
2 <cfscript>
3   r = new coldbox.system.testing.TestBox( directory={
4       mapping = "coldbox.testing.cases.testing.specs",
5       recurse = true,
6       filter = function( path ){ return true; }
7   });
8 </cfscript>
9 <cfoutput>#r.run(reporter="simple")#</cfoutput>
```

```
1 <cfsetting showdebugoutput="false" >
2 <cfparam name="url.reporter" default="simple">
3 <!-- Directory Runner -->
4 <cfset r = new coldbox.system.testing.TestBox( directory={ mapping = "coldbox.testing.cases.testing.specs", recurse = true } ) >
5 <cfoutput>#r.run(reporter=url.reporter)#</cfoutput>
```



RUNNER SAMPLES

```
1 <cfsetting showdebugoutput="false" >
2 <cfscript>
3   r = new coldbox.system.testing.TestBox( directory={
4       mapping = "coldbox.testing.cases.testing.specs",
5       recurse = true,
6       filter = function( path ){ return true; }
7   });
8 </cfscript>
9 <cfoutput>#r.run(reporter="simple")#</cfoutput>
```

```
1 <cfsetting showdebugoutput="false" >
2 <cfparam name="url.reporter" default="simple">
3 <!-- Directory Runner -->
4 <cfset r = new coldbox.system.testing.TestBox( directory={ mapping = "coldbox.testing.cases.testing.specs", recurse = true } ) >
5 <cfoutput>#r.run(reporter=url.reporter)#</cfoutput>
```

```
1 <cfsetting showdebugoutput="false" >
2 <cfparam name="url.reporter" default="simple">
3 <!-- One runner -->
4 <cfset r = new coldbox.system.testing.TestBox( bundles="coldbox.testing.cases.testing.specs.AssertionsTest", labels="railo" ) >
5 <cfoutput>#r.run(reporter="#url.reporter#")#</cfoutput>
```


What is B.D.D.?



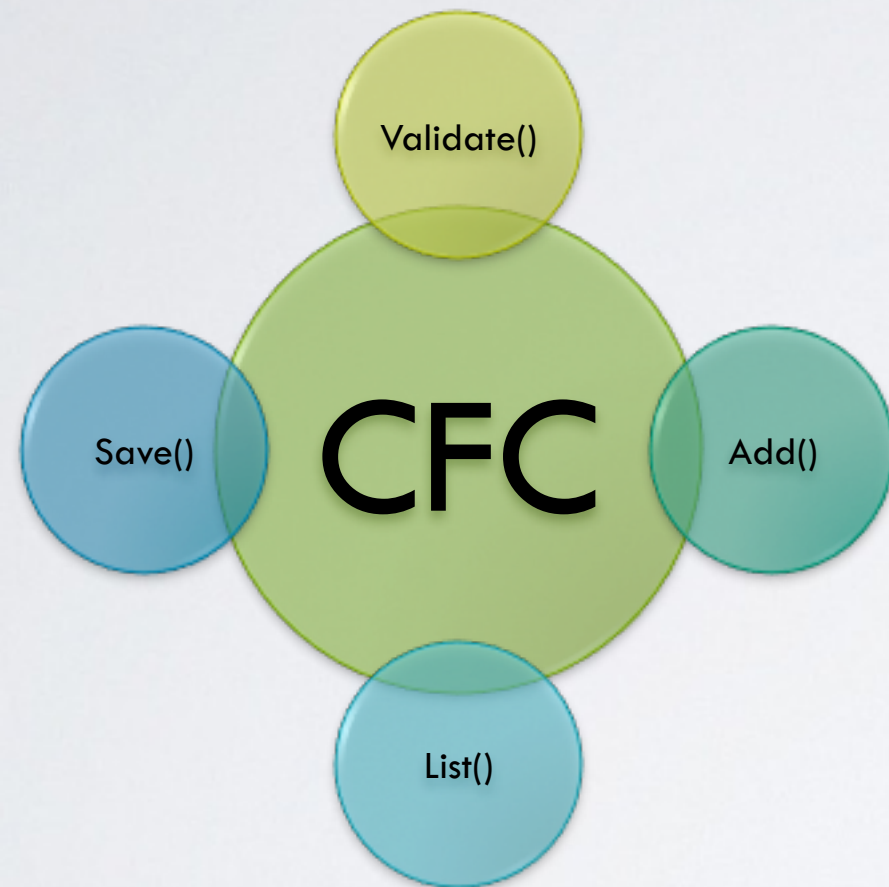
What is B.D.D.?

In software engineering, behavior-driven development (**BDD**) is a software development **process based** on test-driven development (**TDD**). Behavior-driven development combines the general techniques and principles of TDD with ideas from **domain-driven design** and **object-oriented analysis and design** to provide software **developers** and **business analysts** with **shared tools** and a **shared process** to **collaborate** on software development, with the aim of delivering "**software that matters**"

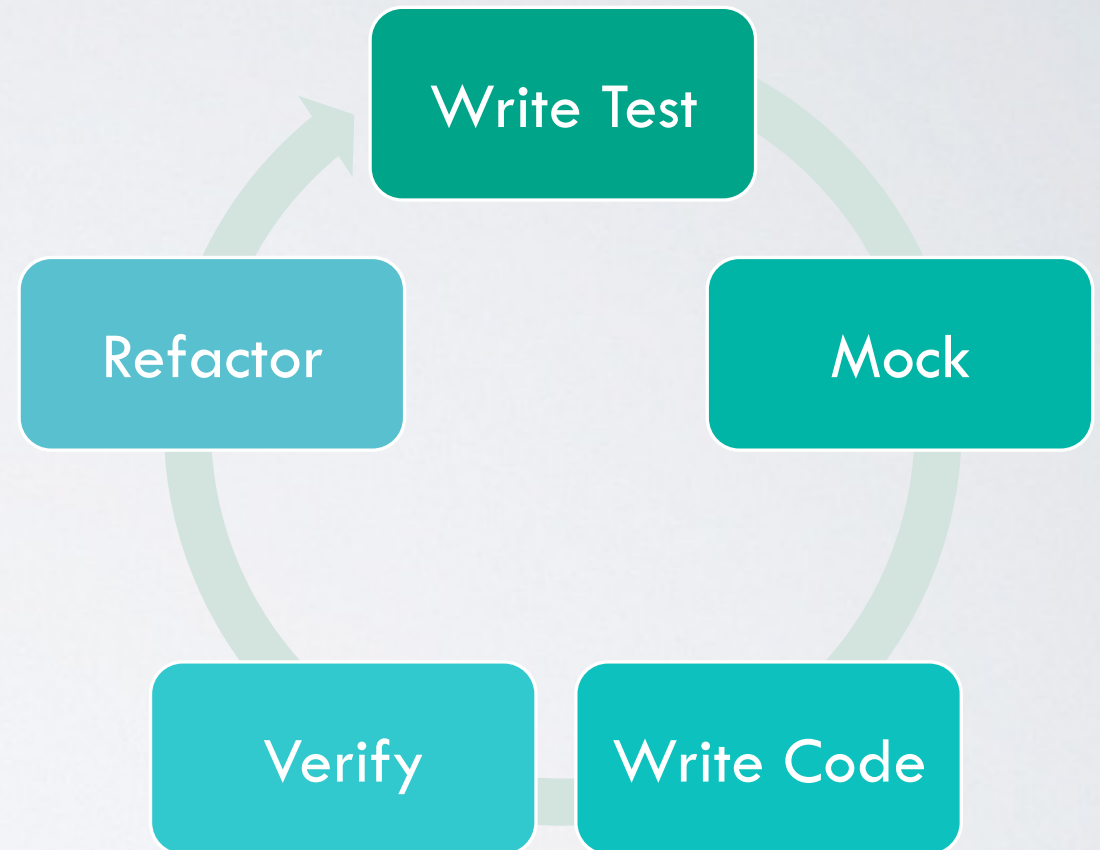
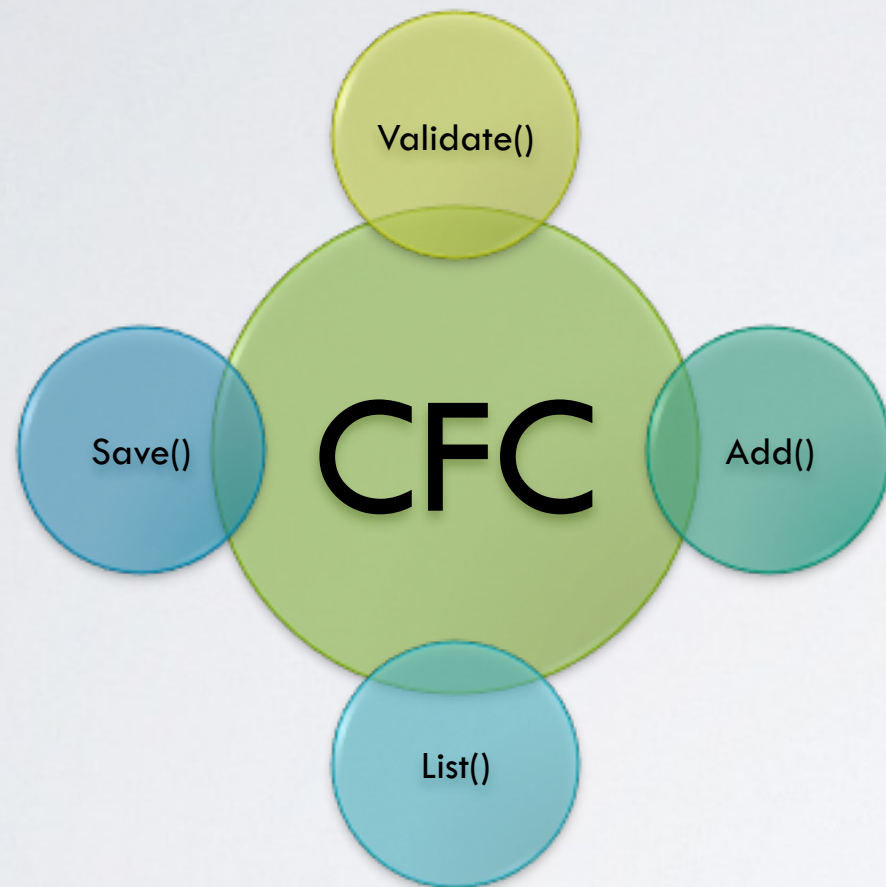


T.D.D.

T.D.D.



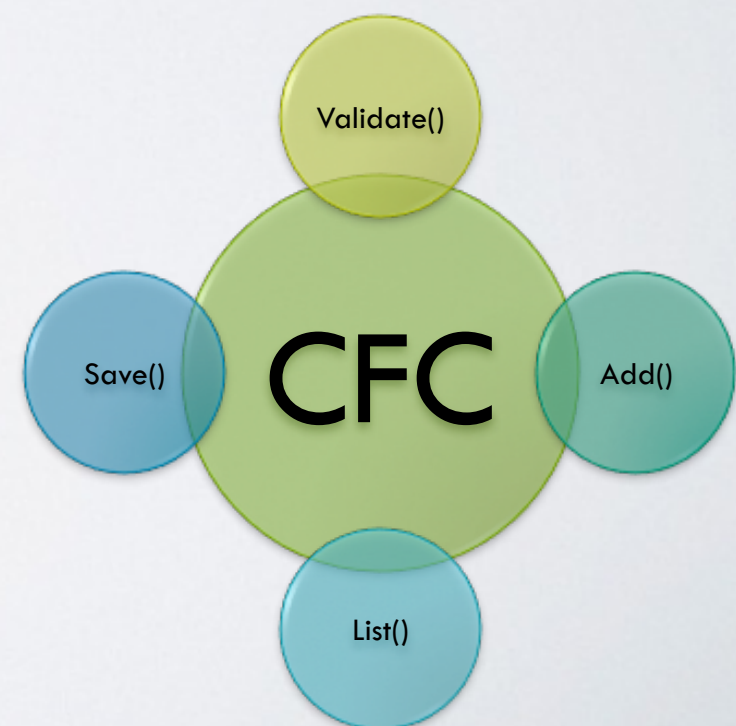
T.D.D.



TDD Process

T.D.D.

- It is an approach to develop software by writing code that exercises your code
- It does help you:
 - Have immediate feedback
 - Create tests before rather than after (yea right!)
 - Express behavior and ideas
 - Creates some documentation
 - Verifies your source code compiles and executes



T.D.D.

- Is not about verifying software requirements
- Does not:
 - Verify user's or stakeholder expectations
 - Express that requirements are satisfied
 - Very very developer oriented
 - Tedious as we always have to test methods and refactoring is a pain
- Let's be truthful, TDD can be a pain in the buttocks!
- We start strong, but we finish weak, even if we finish



T.D.D.

- Is not about verifying software requirements
- Does not:
 - Verify user's or stakeholder expectations
 - Express that requirements are satisfied
 - Very very developer oriented
 - Tedious as we always have to test methods and refactoring is a pain
- Let's be truthful, TDD can be a pain in the buttocks!
- We start strong, but we finish weak, even if we finish



**Developer Test
Paralysis**

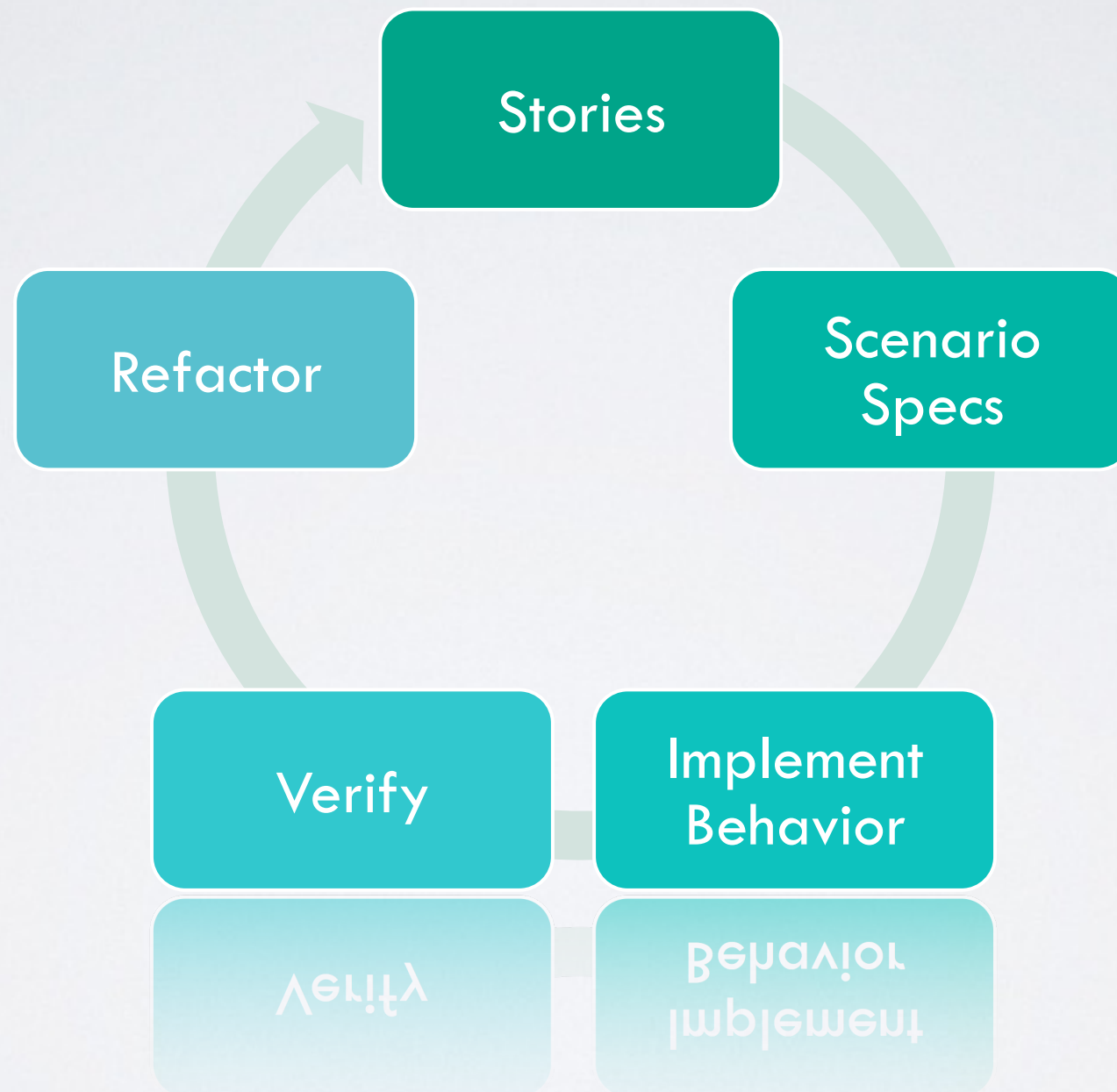


>> B.D.D is T.D.D. Evolved

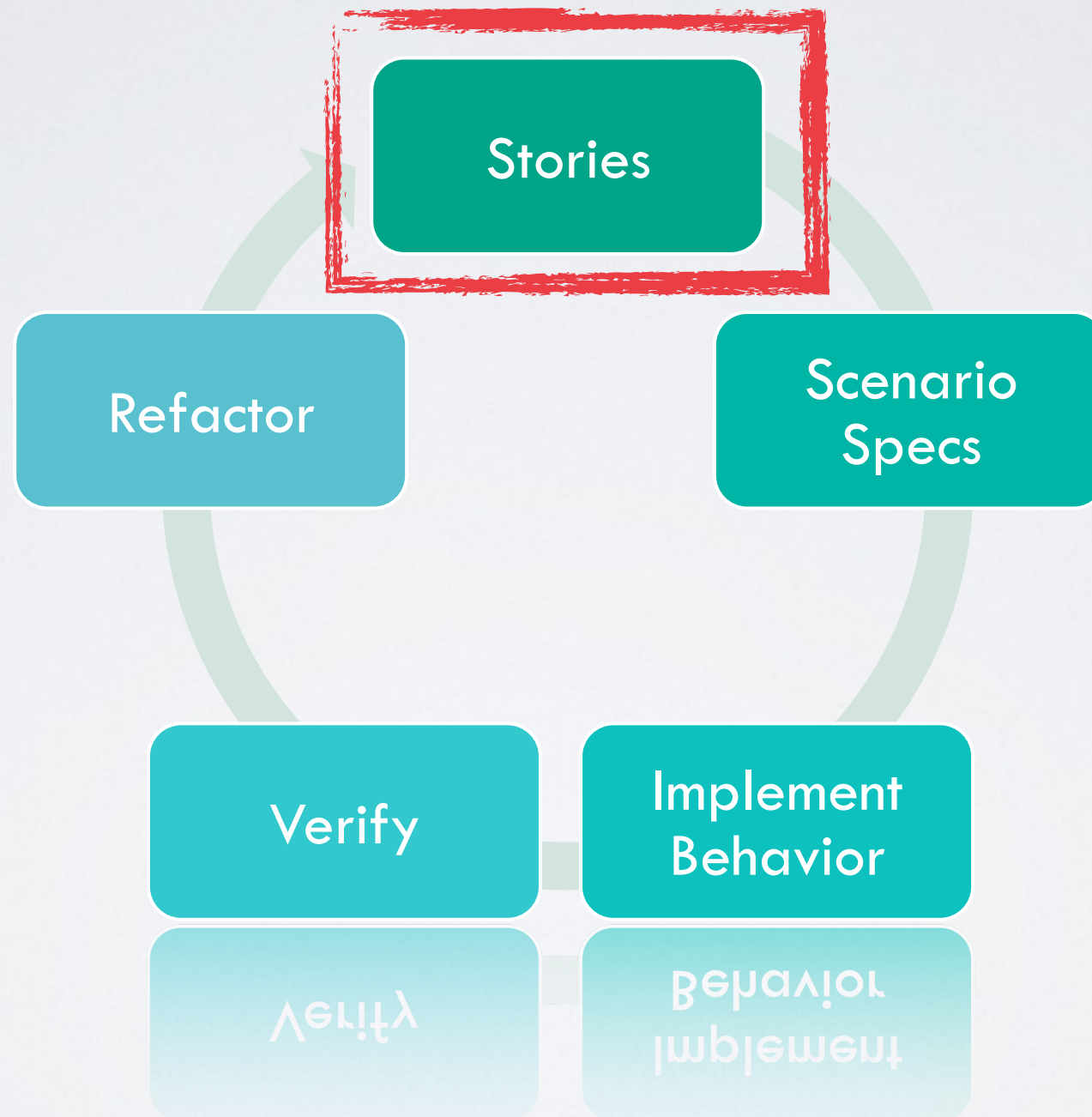
- Dan North
 - <http://dannorth.net/introducing-bdd/>
- Ubiquitous language
 - *existing or being everywhere at the same time : constantly*
- Promotes communication & collaboration between
 - *Developers + business analysts + stakeholders*
- Focuses on **stories** or **requirements** rather than on **functions**
- Focuses on what a system **should** do and not on **how** it should be implemented
- Better readability and visibility
- Verify that software works but also that it meets customer expectations

B.D.D. Process

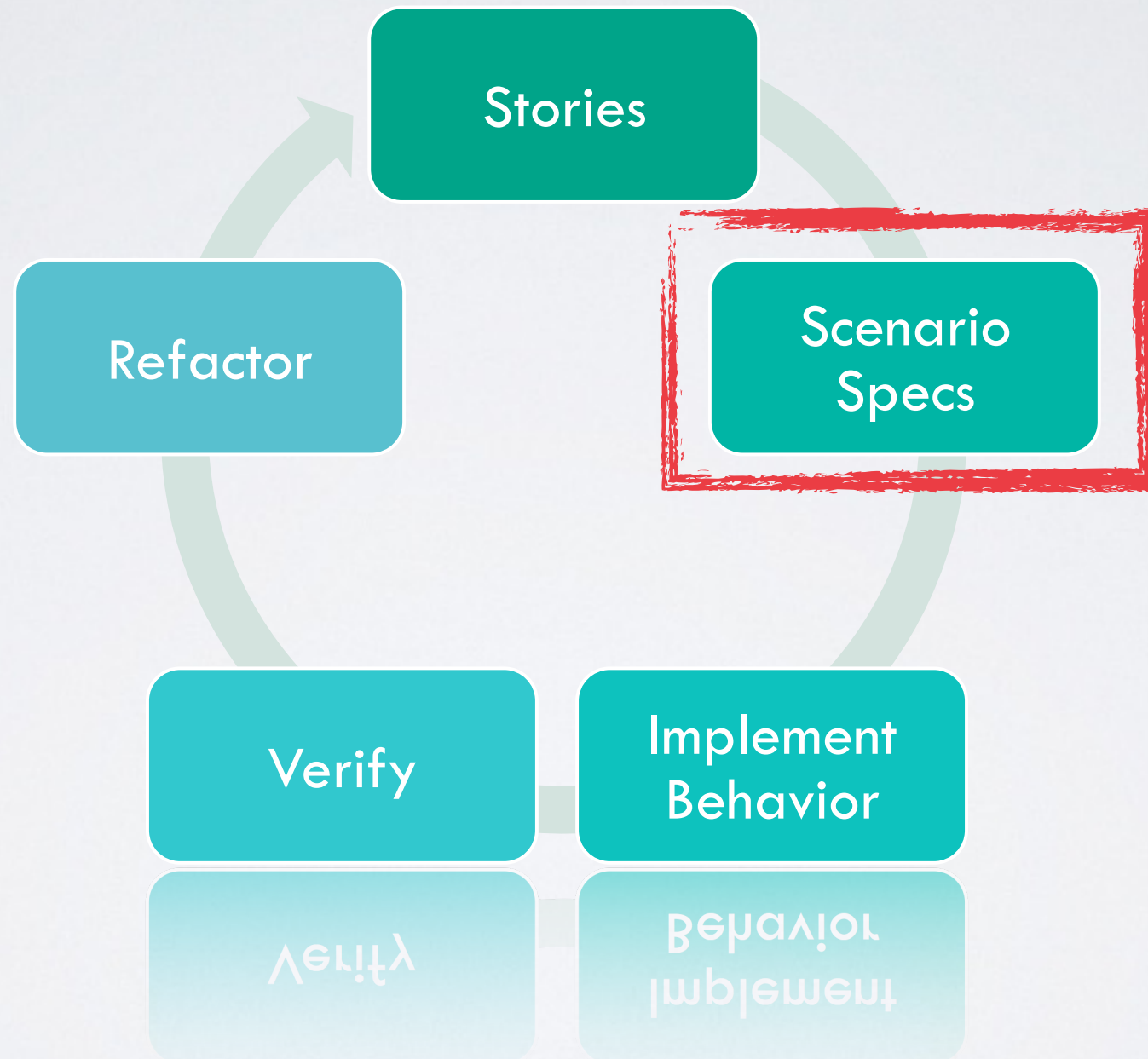
B.D.D. Process



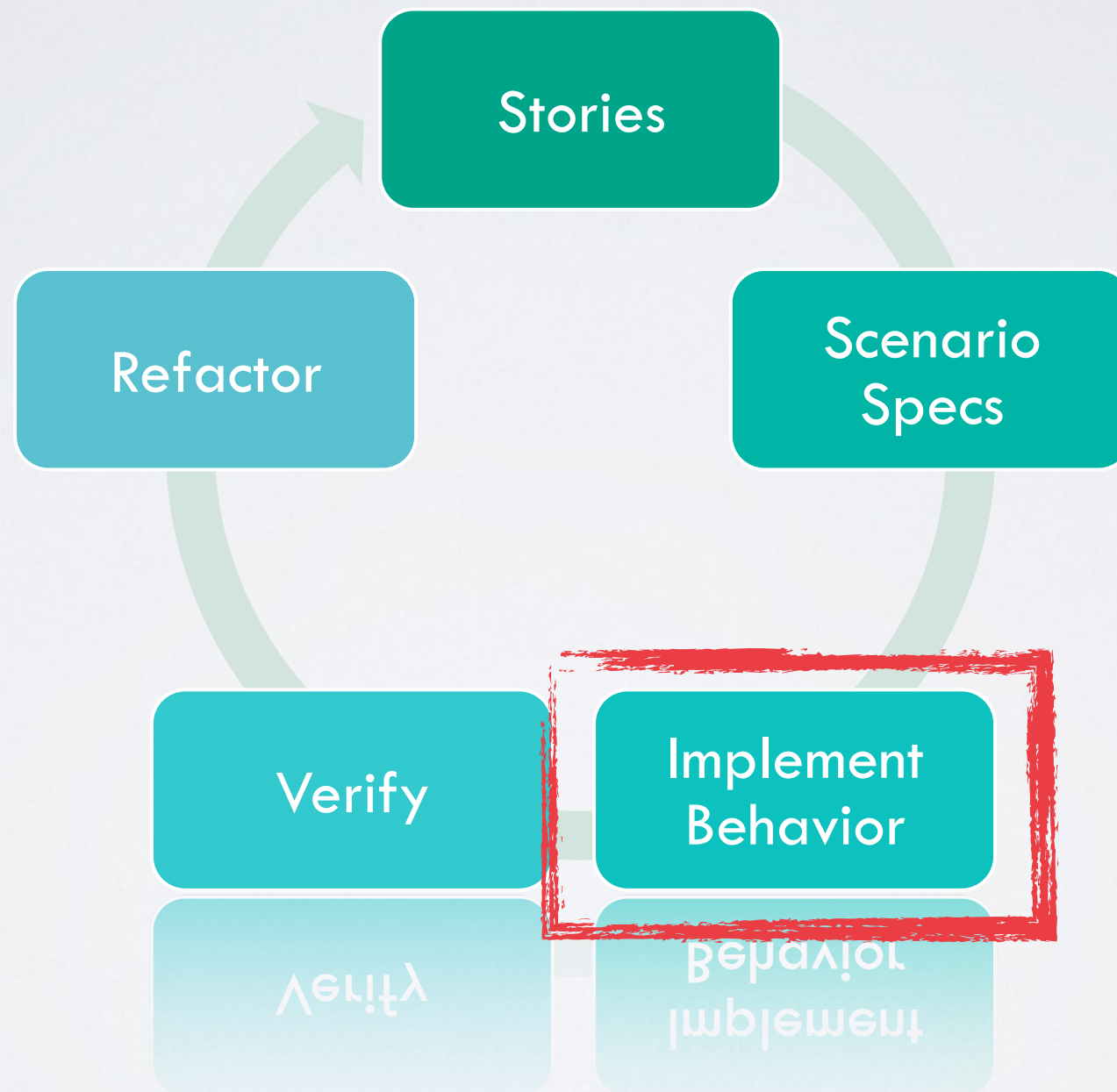
B.D.D. Process



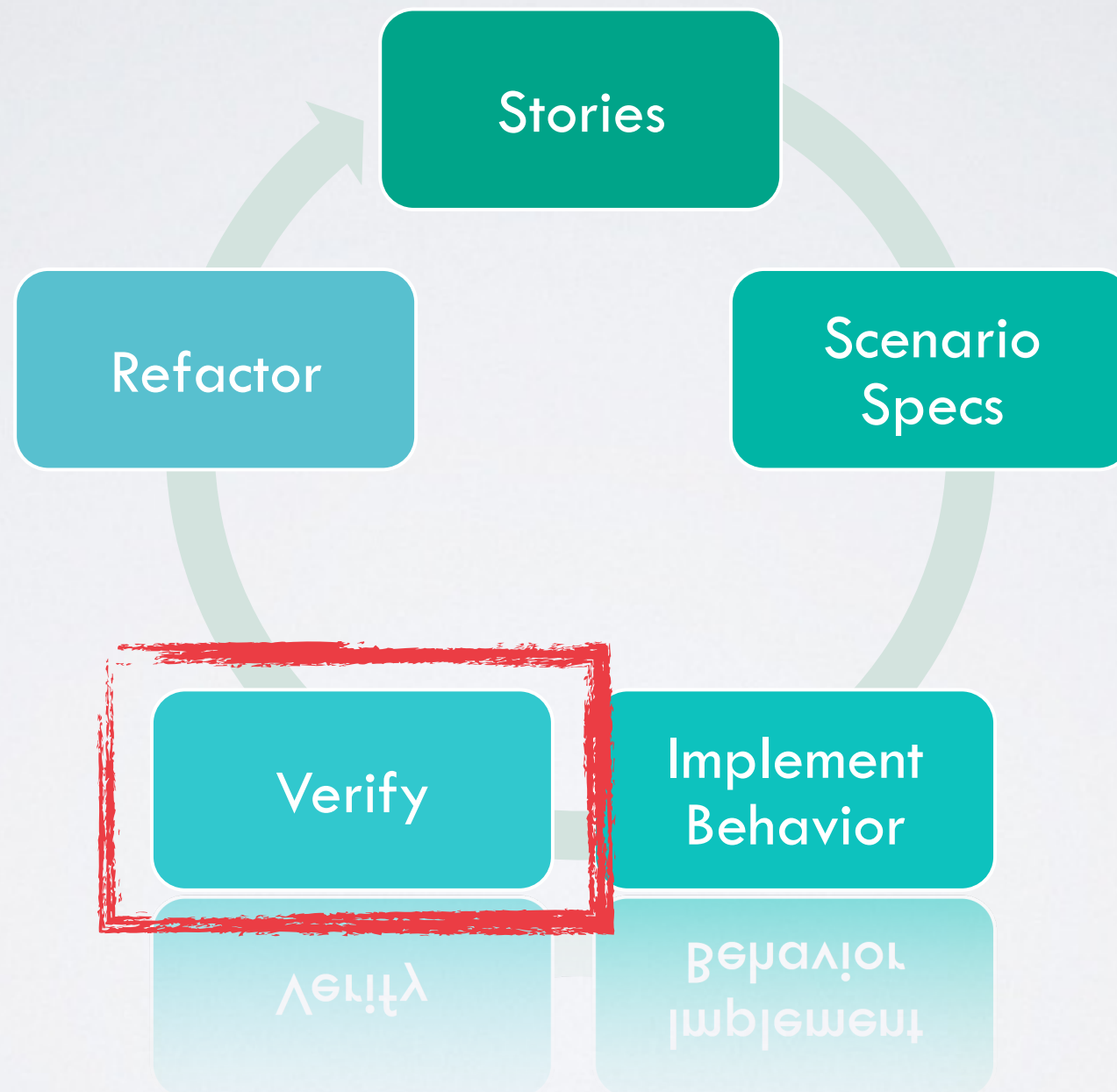
B.D.D. Process



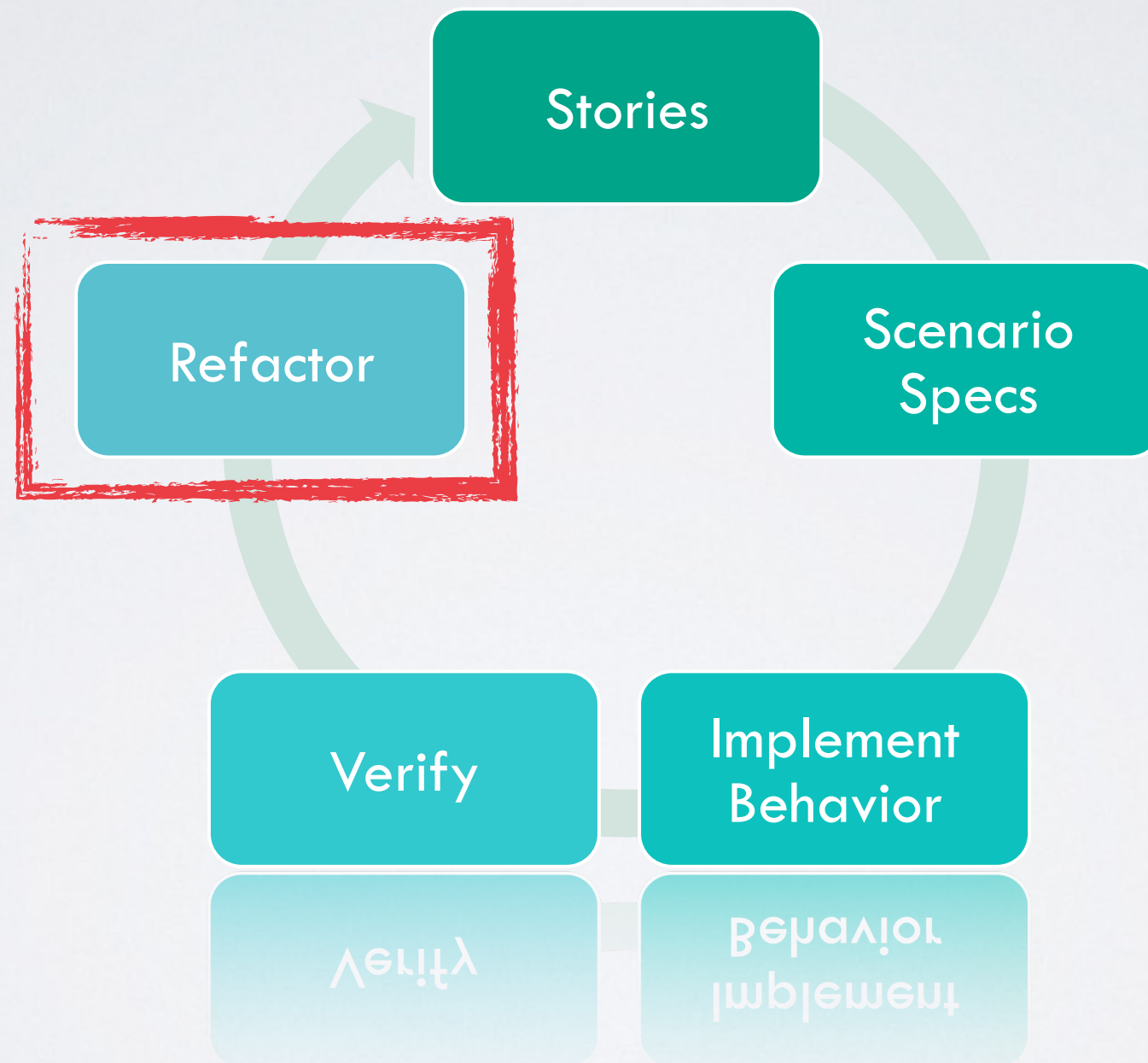
B.D.D. Process



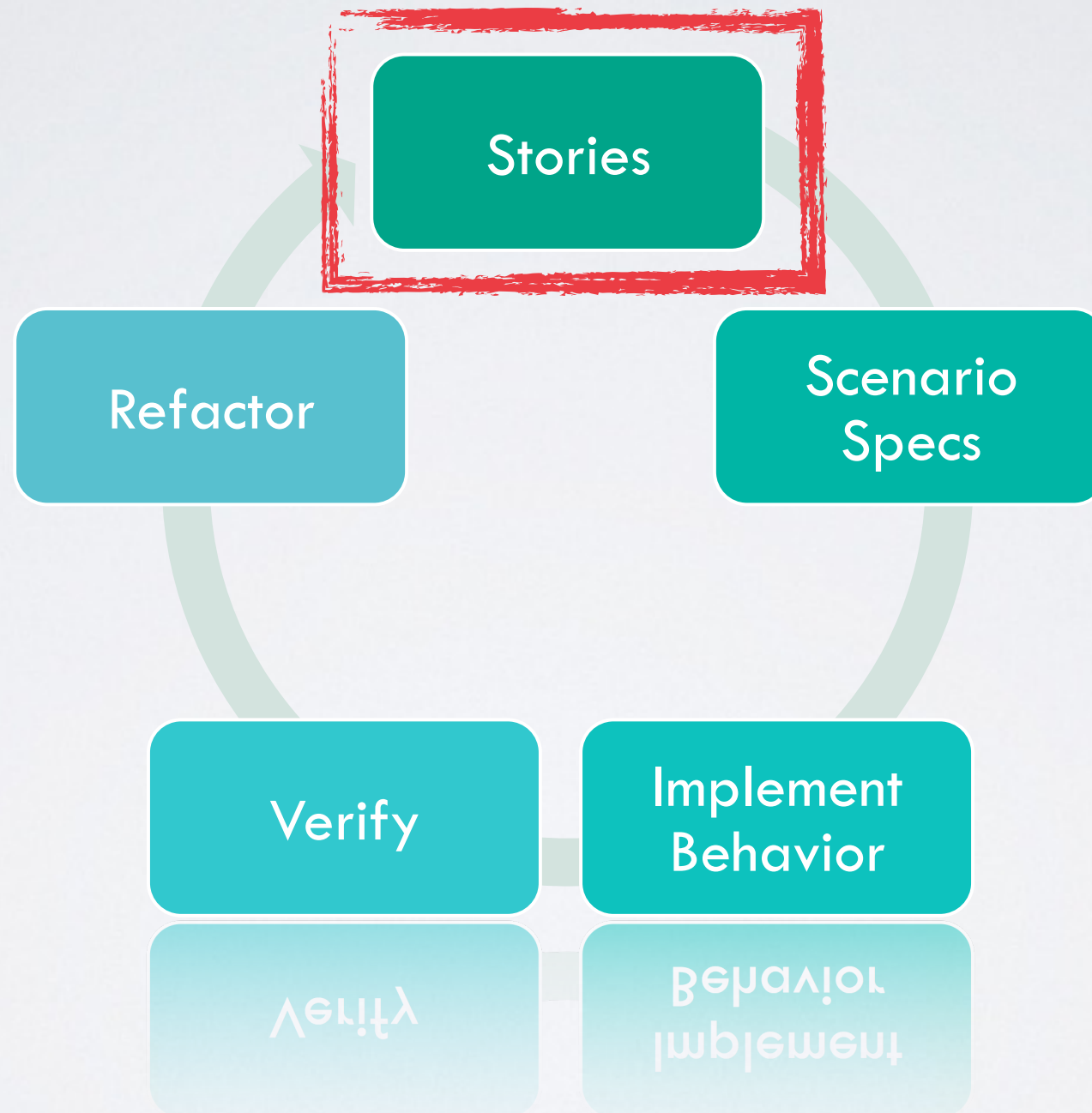
B.D.D. Process



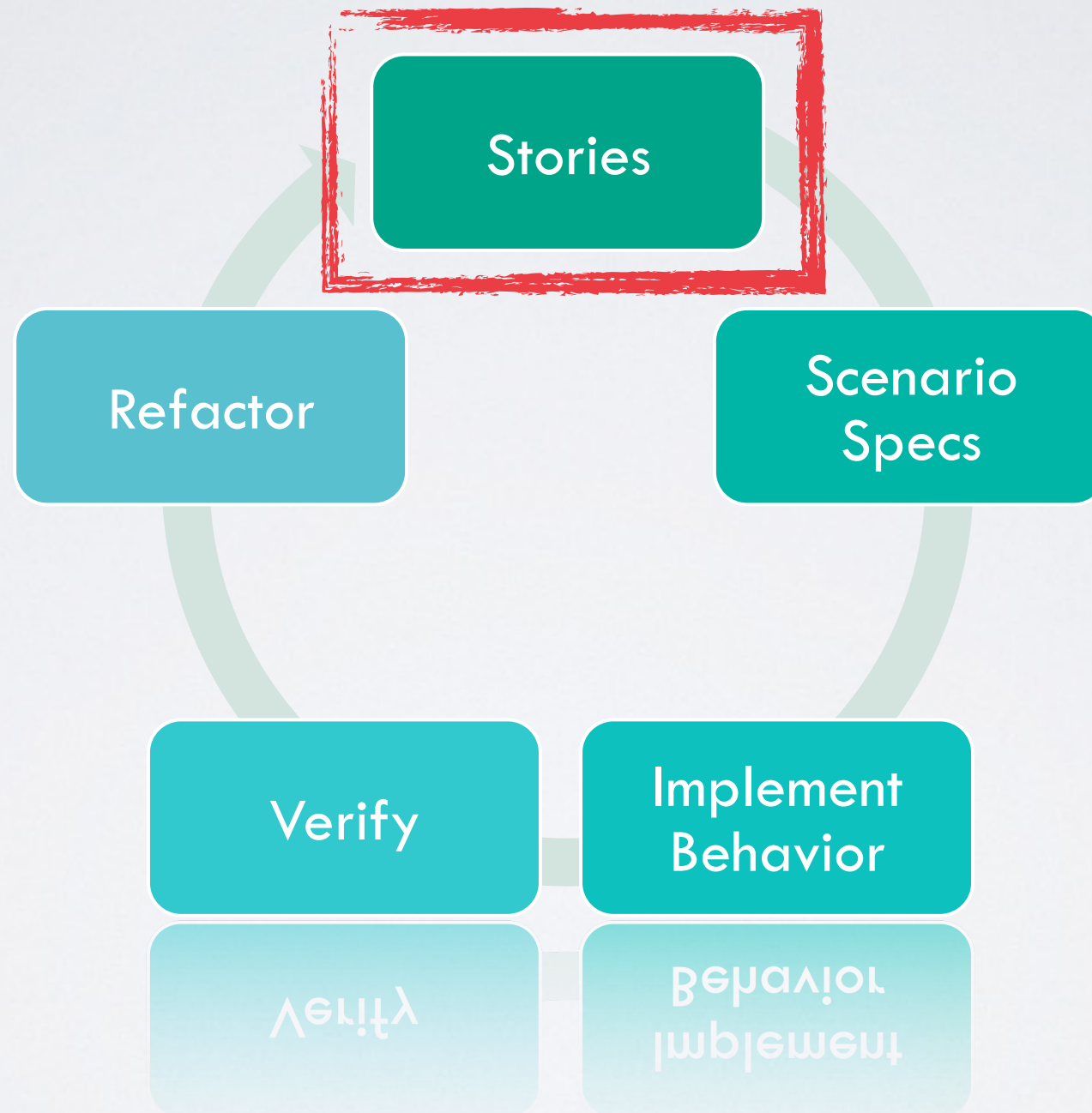
B.D.D. Process



B.D.D. Process



B.D.D. Process







Story Framework

- *Where to start?*
 - *Outside – In*
- *What to test?*
 - *User stories*
- *What not to test?*
 - *Anything else*



Story Framework

- *Where to start?*
 - *Outside – In*
- *What to test?*
 - *User stories*
- *What not to test?*
 - *Anything else*







Stories to Scenarios



Stories to Scenarios

As an application user

I want to be welcomed with my name at login
in order to personalize my experience



Scenario: user login with valid credentials

Given a user "**luis**" with password "**secret**" exists

When I login as "**luis**" with "**secret**"

Then I should see the message "**Welcome back luis!**"



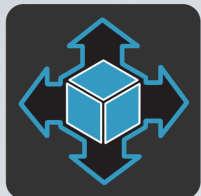


Stories to Scenarios



Stories to Scenarios





TESTBOX



Story to Scenario to TestBox



Story to Scenario to TestBox

As an application user
I want to be welcomed with my name at login
in order to personalize my experience



Scenario: user login with valid credentials
Given a user "luis" with password "secret" exists
When I login as "luis" with "secret"
Then I should see the message "Welcome back luis!"



```
describe( "User login with valid credentials", function() {  
  it( "should see a personalized message", function() {  
    userService.login( "luis", "secret" )  
    var event = execute("user.home")  
    expect( event.getValue( "welcome" ) ).toBe( "Welcome back luis!" )  
  })  
})
```

```
})
```

```
})
```

```
expect( event.getDefault( "message" ) ).toBe( "MESSAGE BACK LUIS!" )
```




TEST BUNDLE CFC

- **run()**
 - Declare your scenario specs + suites
- Life-cycle methods
- Inherit some assertion, utility and mocking methods
- Expectations Library: **expect()**
- Assertions Library: **\$assert**

```
/**
 * My BDD Test
 */
component extends="testbox.system.testing.BaseSpec"{

    /******* LIFE CYCLE Methods *****/

    // executes before all suites+specs in the run() method
    function beforeAll(){
    }

    // executes after all suites+specs in the run() method
    function afterAll(){
    }

    /******* BDD SUITES *****/

    function run(){
        // all your suites go here.
    }
}

}

// all your suites go here.
function run(){
```



SUITES: DESCRIBE() YOUR TESTS

- Suites begin with a **describe()** block
 - title
 - closure
- A suite is composed of specs or more suites
- Closures can contain
 - Life-cycle methods
 - More suites
 - Specs

```
describe("A suite", function(){
  beforeEach(function( currentSpec ){
    obj = new myObj();
  });
  afterEach(function( currentSpec ){
    structClear( session );
  });
  it("contains spec with an awesome expectation", function(){
    expect( true ).toBeTrue();
  });
});
```

```
});
});
expect( false ).toBeFalse();
it("contains spec with an awesome expectation", function(){
```



DESCRIBE() ARGUMENTS

Annotation	Description
Title	The title of the suite to register
Body	The closure that represents the test suite
Labels	The list or array of labels this suite group belongs to
Skip	A flag or a closure that tells TestBox to skip this suite group from testing if true. If this is a closure it must return boolean.
AsyncAll	If you want to parallelize the execution of the defined specs in this suite group.



SPECS: IT()

- At least 2 args
 - **Title**
 - **Closure**
 - Labels
 - Skip
- Closure is where you define
 - Scenarios
 - **1+ expectations**, assertions
- Specs tested in order declared

```
describe("A suite", function(){  
  it("contains spec with an awesome expectation", function(){  
    expect( true ).toBeTrue();  
  });  
  it("contains a spec with more than 1 expectation", function(){  
    expect( [1,2,3] ).toBeArray();  
    expect( [1,2,3] ).toHaveLength( 3 );  
  });  
});
```

```
});  
});  
expect( [1,2,3] ).toHaveLength( 3 );
```



EXPECTATIONS + MATCHERS

- Self-concatenated method calls that evaluate your **SUT**
- Start with a call to **expect(actual)**
- Concatenated with **matcher** methods
 - TestBox ships with a plethora of matchers!
- Matchers also have negation (**not**) counterparts

```
describe("A suite", function(){
  it("contains a spec with more than 1 expectation", function(){
    expect( true ).toBeTrue();
    expect( false ).notToBeTrue();
    expect( [1,2,3] ).toBeArray();
    expect( [1,2,3] ).toHaveLength( 3 );
  });
});

});
});
expect( [1,2,3] ).toHaveLength( 3 );
```



NESTED SUITES

- Nest to your heart's delight
- Life-cycle methods bubble up
- Great for grouping and recursive scenarios
- Execute in descending order

```
describe( "View Operations", function(){  
  it( "can get a view", function(){  
    var view = couchbase.getView( 'beer', 'by_location' );  
    expect( view.getViewName() ).toBe( 'by_location' );  
  });  
  
  describe( "Validate Query Options", function(){  
    it( "can throw error on invalid stale option", function(){  
      expect( function(){  
        couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: 20, stale: 'invalid' } );  
      }).toThrow( type="InvalidStale" );  
    });  
  
    it( "can throw error on invalid limit", function(){  
      expect( function(){  
        couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: -5 } );  
      }).toThrow( type="InvalidLimit" );  
      expect( function(){  
        couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: 'invalid' } );  
      }).toThrow( type="InvalidLimit" );  
    });  
  });  
});  
_;
```




EXPECTING EXCEPTIONS

- Verifies exceptions
- Pass a closure to expect() and use toThrow(type, regex)
- Match types and message+detail regex

```
it( "can throw error on invalid stale option", function(){
  expect( function(){
    couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: 20, stale: 'invalid' } );
  }).toThrow( type="InvalidStale" );
});

it( "can throw error on invalid limit", function(){
  expect( function(){
    couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: -5 } );
  }).toThrow( type="InvalidLimit" );
  expect( function(){
    couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: 'invalid' } );
  }).toThrow( type="InvalidLimit" );
});
```

```
});

it( "can throw error on invalid limit", function(){
  expect( function(){
    couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: -5 } );
  }).toThrow( type="InvalidLimit" );
  expect( function(){
    couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: 'invalid' } );
  }).toThrow( type="InvalidLimit" );
});
```



LABELS + SKIPPING

- Apply labels to suites and specs
- Skip suites and specs
- Shortcut methods: **xDescribe()**, **xit()**

```
describe("A suite", function(){
  xit("contains a spec with more than 1 expectation", function(){
    expect( true ).toBeTrue();
  });
});

xdescribe("A skipped suite", function(){
  it("contains a spec with more than 1 expectation", function(){
    expect( true ).toBeTrue();
  });
});

describe( "View Operations", function(){
  it( "can get a view", function(){
    var view = couchbase.getView( 'beer', 'by_location' );
    expect( view.getViewName() ).toBe( 'by_location' );
  });
});

});
});

expect( view.getViewName() ).toBe( 'by_location' );
var view = couchbase.getView( 'beer', 'by_location' );
```




ASYNCHRONICITY

- TestBox executes all specs in parallel
- Decided on a per **describe()** block
- Attention to var-scoping
- Test shared access and contention
- Much power comes much responsibility!

```
describe( title="Validate Query Options", asyncAll=true, body=function(){
  it( "can throw error on invalid stale option", function(){
    expect( function(){
      couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: 20, stale: 'invalid' } );
    }).toThrow( type="InvalidStale" );
  });
  it( "can throw error on invalid limit", function(){
    expect( function(){
      couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: -5 } );
    }).toThrow( type="InvalidLimit" );
    expect( function(){
      couchbase.query( designDocumentName='beer', viewName='brewery_beers', options={ limit: 'invalid' } );
    }).toThrow( type="InvalidLimit" );
  });
};
};
};
```


What the future holds!

- Grunt JS Tasks
- NodeJS Runners and Watchers
- CPU Runners for CF Builder
- More Sublime Integration
- Eclipse Runners
- More Reporters
- More focus on automation
- Gherkins support

What the future holds!

- Grunt JS Tasks
- NodeJS Runners and Watchers
- CPU Runners for CF Builder
- More Sublime Integration
- Eclipse Runners
- More Reporters
- More focus on automation
- Gherkins support

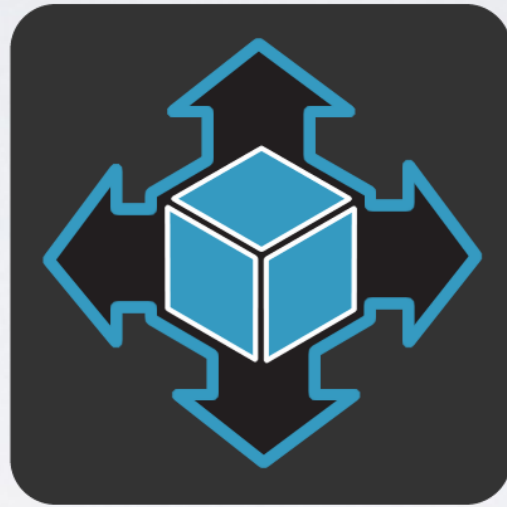


TESTBOX

Q & A

Thanks!

Q & A



TESTBOX

TESTBOX

Thanks!