Test-Driven Development

by Craig L. Jones

craig@ChiefSimplicityOfficer.com

30 Years Software Development

12 Years Agile
(mostly XP, Scrum, & Kanban)

CSM & Scrum Coach

Recovering Over-engineer-er

www.ChiefSimplicityOfficer.com

Your Expectations for Tonight?

As a _____(role)_____

I want TDD in my tool box

So that _____(expectation)_____

Wax On
Wax Off

Sand the Floor

Paint the Fence

Automatic Reflex

What development practices ought to be ingrained in muscle memory?

1. _____
2. _____
3. _____
4. _____

Automatic Reflex

What development practices ought to be ingrained in muscle memory?

Craig's Nominations:

1. Frequent Demos & Feedback Gathering
2. Test-Driven Development
3. Continuous Integration & Delivery
4. Tracking Technical Debt and being able to articulate it in actual dollars

Why Test?

Why Test?

- Primarily: Ensure that we deliver value -- that we collect on our investment
- Secondarily: Safety, Privacy, Security
- And in no small measure: Pride of Work, Peace of Mind

Why Automate Tests?

Why Automate Tests?

- Allows **refactoring** while preserving functionality
- Avoids having to break out the **debugger**
- **Regression testing** on demand
- **“Executable documentation”** (reference implementation, exploratory testing)

Challenges w/Automated Tests?

Challenges w/Automated Tests?

- Writing setup code can be tedious
- Keeping the tests current can be hard
- Allowing for random factors isn't easy (arbitrary order or array elements, timestamps, time zones, user settings)

Why Test-First?

Why Test-First?

1. Guarantees that the code is testable.

- Testable code is well-designed code. (TDD almost forcefully injects the First Principles into your code.)

Why Test-First?

2. It clarifies the requirements.

- Vagueness gets nailed down
- Latent requirements are surfaced.

Why Test-First?

3. It clarifies the programming task.

- Particularly with an emphasis on the (abstract) interfaces.

- Allows for blazing-bright focus on that task.

- No doubt about when the task is done.

Why Test-First?

4. The tools that generate code from the tests are much better than the tools that generate tests from the code.

(more on this in the Tips section, below)

Why Test-First?

In other words...

“Code to the test.”

(Is that “cheating?”)

The TDD Process

1. Write a (failing) test.
2. Write (just enough) production code to make the test pass. (All of the previously written tests must still pass, too.)
3. Think of another test and repeat.
4. Keep going, until you can't think of any more tests.

The TDD Process

1. Write a (failing) test.
2. Write (just enough) production code to make the test pass. (All of the previously written tests must still pass, too.)
3. Refactor the production code. ("Clean Code")
4. Refactor the test code.
5. Think of another test and repeat.
6. Keep going, until you can't think of any more tests.

The TDD Process

The 3 Laws of TDD, according to “Uncle Bob”

1. You are not allowed to write any production code unless it is to make a failing test pass.
2. You are not allowed to write any more of a test than is sufficient to fail; and compilation failures are failures.
3. You are not allowed to write any more production code than is sufficient to pass the one failing unit test.

Test-Driven Development

(Team Exercise)

Test-Driven Limericks

A Classic Limerick...

There once was a sailor from Wales
Who was expert at pissing in gales
    From the uppermost spar
    He could fill a small jar
And not get any on the sails

Test-Driven Limericks

A new Limerick, written test-driven...

A young superhero named Bellows
Shot heat rays in bright reds, and yellows
    The Ice Queen he married
    O'r the threshold he carried
And then, ... Wow! What steamy bed fellows!

Test-Driven Limericks

Let's write another...

Test-Driven Limericks

Test #1: A limerick tells a consistent story:

Test-Driven Limericks

An accountant took cooking lessons at night.

Test-Driven Limericks

Test #1: A limerick tells a consistent story:

Test #2: A limerick story usually has a twist:

Test-Driven Limericks

An accountant took cooking lessons at night, but, he wasn't very careful and ended up cooking the books.

Test-Driven Limericks

Test #1: A limerick tells a consistent story:
Test #2: A limerick story usually has a twist:
Test #3: A limerick always consists of 5 lines.

Test-Driven Limericks

An accountant took cooking lessons at night.
There was a lot of homework.
He tried to do both jobs at once.
He lost focus and got mixed up.
He ended up cooking the books.

Test-Driven Limericks

Test #1: A limerick tells a consistent story:

Test #2: A limerick story usually has a twist:

Test #3: A limerick always consists of 5 lines.

Test #4: The 1st line usually either names the protagonist, or names where the protagonist lives.

Test-Driven Limericks

An accountant **named Bob** took cooking lessons at night.
There was a lot of homework.
He tried to do both jobs at once.
He lost focus and got mixed up.
He ended up cooking the books.

Test-Driven Limericks

Test #1: A limerick tells a consistent story:
Test #2: A limerick story usually has a twist:
Test #3: A limerick always consists of 5 lines.
Test #4: The 1st line usually either names the protagonist, or names where the protagonist lives.
Test #5: The 1st line traditionally starts with "There once was a...", and ends with the named protagonist or locale.

Test-Driven Limericks

There once was an accountant named Bob. **//**
He took cooking lessons at night.
There was a lot of homework.
He tried to do both jobs at once, **~** lost focus and got mixed up.
He ended up cooking the books.

// = had to split the line
~ = had to combine lines to keep to 5 lines

Test-Driven Limericks

Test #1: A limerick tells a consistent story:
Test #2: A limerick story usually has a twist:
Test #3: A limerick always consists of 5 lines.
Test #4: The 1st line usually either names the protagonist, or names where the protagonist lives.
Test #5: The 1st line traditionally starts with "There once was a...", and ends with the named protagonist or locale.
Test #6: The 1st line must contain 3 stressed syllables.

Test-Driven Limericks

There **ONCE** was an **AC**-count-ant named **BOB**.
[Already good]
He took cooking lessons at night.
There was a lot of homework.
He tried to do both jobs at once, lost focus and got mixed up.
He ended up cooking the books.

Test-Driven Limericks

Test #1: A limerick tells a consistent story:
Test #2: A limerick story usually has a twist:
Test #3: A limerick always consists of 5 lines.
Test #4: The 1st line usually either names the protagonist, or names where the protagonist lives.
Test #5: The 1st line traditionally starts with "There once was a...", and ends with the named protagonist or locale.
Test #6: The 1st line must contain 3 stressed syllables.
Test #7: The 2nd line must also have 3 stressed syllables.

Test-Driven Limericks

There once was an accountant named Bob
He **TOOK COOK**-ing **LES**-sons at **NIGHT**. [4]

||

V

Who was **LEARN**-ing to **COOK** after **WORK**. [3]

Test-Driven Limericks

Test #1: A limerick tells a consistent story:
Test #2: A limerick story usually has a twist:
Test #3: A limerick always consists of 5 lines.
Test #4: The 1st line usually either names the protagonist, or names where the protagonist lives.
Test #5: The 1st line traditionally starts with "There once was a...", and ends with the named protagonist or locale.
Test #6: The 1st line must contain 3 stressed syllables.
Test #7: The 2nd line must also have 3 stressed syllables.
Test #8: The 2nd line rhymes with the first.

Test-Driven Limericks

There once was an accountant named **Bob**
He took cooking lessons at night.

There was an accountant named **Jeff**
Who was also an amateur chef

Test-Driven Limericks

Test #1: A limerick tells a consistent story:
Test #2: A limerick story usually has a twist:
Test #3: A limerick always consists of 5 lines.
Test #4: The 1st line usually either names the protagonist, or names where the protagonist lives.
Test #5: The 1st line traditionally starts with "There once was a...", and ends with the named protagonist or locale.
Test #6: The 1st line must contain 3 stressed syllables.
Test #7: The 2nd line must also have 3 stressed syllables.
Test #8: The 2nd line rhymes with the first.
Test #9: The 3rd and 4th sentences each have TWO stressed syllables, and they rhyme.

Test-Driven Limericks

There was a lot of homework.
He tried to do both jobs at once, lost focus and got mixed up.

Sadly, it looks
Like he cooked the wrong books

Test-Driven Limericks

Test #1: A limerick tells a consistent story:
Test #2: A limerick story usually has a twist:
Test #3: A limerick always consists of 5 lines.
Test #4: The 1st line usually either names the protagonist, or names where the protagonist lives.
Test #5: The 1st line traditionally starts with "There once was a...", and ends with the named protagonist or locale.
Test #6: The 1st line must contain 3 stressed syllables.
Test #7: The 2nd line must also have 3 stressed syllables.
Test #8: The 2nd line rhymes with the first.
Test #9: The 3rd and 4th sentences each have TWO stressed syllables, and they rhyme.
Test #10: The 5th sentence has 3 stressed syllables and rhymes with the 1st and 2nd.

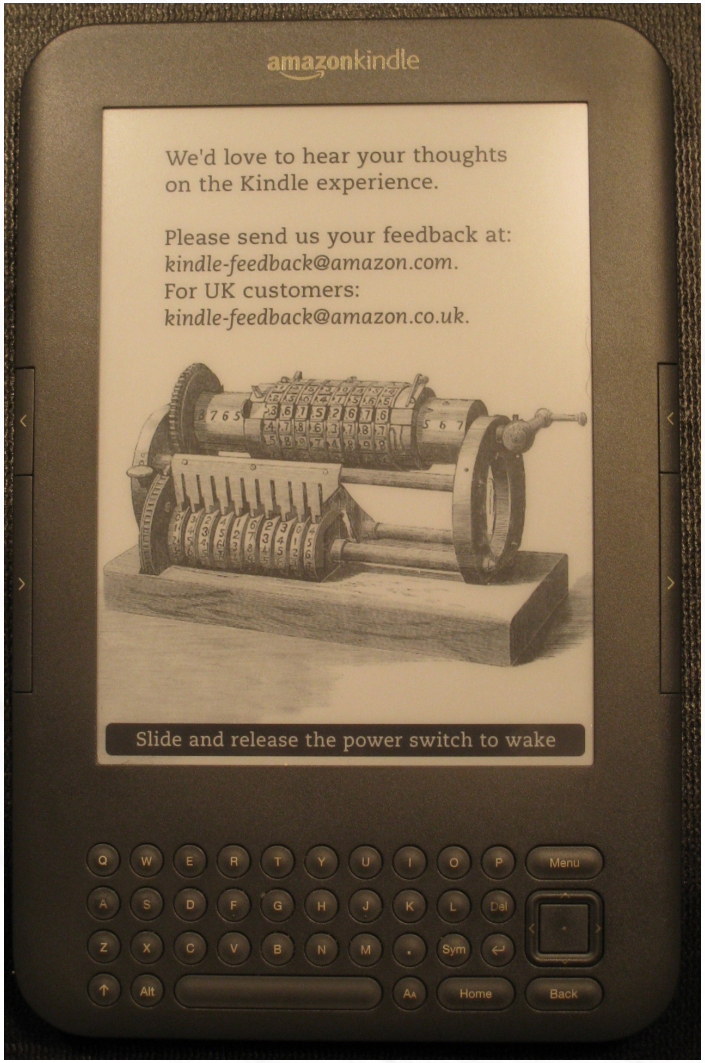
Test-Driven Limericks

There was an accountant named **Jeff**
Who was also an amateur **chef**
    sadly, it looks
    like he cooked the wrong books
and now he has no cabbage **left**

Pair Programming

- Collaboration between two programmers
- Collaboration between programmer & tester
- Collaboration between business analyst & tester
- Collaboration between programmer & business analyst (as tester)

Stories About the Kindle



Basic Story Template

- As a (role)
- I want (request)
- So that (reason)

- In order to (reason)
- As a (role)
- I want (request)

Example User-Story

Feature: Tagging Notes
In order to improve the usefulness of the Kindle system's note-taking ability
As a studious reader who averages 75 highlights and/or notes per e-book
I want the ability to "tag" my highlights/notes with one- or two-word phrases of my choosing (e.g. 'action item', 'vocabulary', 'key thought', 'errata', or 'obsolete')

Basic Acceptance-Test Template

- Pronounced “Ga-Wa-Ta”

- Given
- And
- And
- When
- And
- And
- Then
- And
- And

Example Acceptance-Test

Scenario: Create a new tag
Given I have begun to enter a new note, Or I have begun to edit an existing note
And the note does not yet have a tag
When I navigate to the (new) tag field and begin to enter a word or phrase
Then whatever I type will be associated with that passage
And whatever I type will be added to the list of known tags

Example Bug Report

Bug: Highlighting a Passage that Contains a Word-Wrapped Hyphen

In order for the Kindle to work as designed

As any Kindle user

I want the highlight feature to work properly when the passage I'm highlighting contains a hyphenated word and that word is wrapped at the hyphen.

Example Acceptance-Test

Scenario: Hard Hyphen, Display of Highlighting In Progress
Given I am reading an e-book
And I navigate to a section that contains a hyphenated word
And I adjust the font size until the word-wrapping causes the hyphenated word to split across lines
When I begin to highlight a passage that includes the hyphenated word
Then the inverted characters that track the highlighting should not skip a line.

Testing Technologies

xUnit

JUnit, TestNG, CPPUnit, csUnit, NUnit, JSUnit, PHPUnit, HTTPUnit, FlexUnit

JUnit Example

```
/** Tests adding an item to the cart. */
public void testAddItem() {
  Product book2 = new Product("Moby Dick",
      12.95);
  cart.addItem(book2);
  double expectedBalance = book1.getPrice()
      + book2.getPrice();
  assertEquals(expectedBalance,
      cart.getBalance(), 0.0);
  assertEquals(2, cart.getItemCount());
}
```

Testing Technologies

FIT
Framework for Integration Testing
FitNesse (Java)

Other implementations:
.NET, Python, Ruby, Smalltalk

FitNesse Example

eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	~=3.14
9	3	<5
11	2	4<_<6
100	4	33

```
|eg.Division|
|numerator|denominator|quotient?|
|10       |2          |5        |
|12.6     |3          |4.2      |
|100      |4          |33       |
```

Testing Technologies

Selenium

- All Major Browsers incl. Chrome, Android, iPhone
- Remote-Controllable from xUnit
- Version 2.0 WebDriver API

Selenium 2.0 Example

```
// send test message
driver.findElement(By.id("AutoCompleteTo$InputBox"))
      .sendKeys(to);
driver.findElement(By.id("fSubject")).sendKeys(subject);
driver.switchTo().frame("UIFrame.1");
driver.findElement(By.xpath("//body")).sendKeys(message);
driver.switchTo().frame("UIFrame");
driver.findElement(By.id("SendMessage")).click();
assertEquals(driver.findElement(By.cssSelector(
       "h1.SmcHeaderColor")).getText(),
       "Your message has been sent");
```

Testing Technologies

Cucumber

- Cucumber (runs in Ruby or JRuby)
- Works with Ruby, Java, .NET, Flex or web applications written in any language
- Gherkin (the language)

Cucumber Example

Look familiar?

```
Feature: Search courses
  In order to ensure better utilization of courses
  Potential students should be able to search for
      courses

  Scenario: Search by topic
    Given there are 240 courses which do not have
       the topic "biology"
    And there are 2 courses A001, B205 that each
       have "biology" as one of the topics
    When I search for "biology"
    Then I should see the following courses:
       | Course code |
       | A001        |
       | B205        |
```

Testing Technologies

English

- Manual testing (when all else fails)

Test-First Tips

Use Live Templates to Generate the xUnit Code

Eclipse:

- Window > Preferences > Java > Editor > Templates to define a template
- Ctrl-Space to invoke

IntelliJ:

- Window > Preferences > Java > Editor > Templates to define a template
- File > New Junit, or File > New Gunit to invoke

A JUnit Template

```
package ${PACKAGE_NAME};
#parse("File Header.java")
class ${TESTED_CLASS_NAME}Tests extends TestCase {
#if (${TESTED_CLASS_NAME} != "")
   ${TESTED_CLASS_NAME} ${TESTED_CLASS_NAME};
#end
    void setUp() {
        super.setUp();
        ${TESTED_CLASS_NAME} = new ${TESTED_CLASS_NAME}();
     }

    void tearDown() {
        super.tearDown();
    }

    void testSomething() throws Exception {
    assertEquals("Expected",${TESTED_CLASS_NAME}.something())
    }
}
```

A JUnit Template

```
package Accounting;

class AccountTransactionTests extends TestCase {
   AccountTransaction accountTransaction;

    void setUp() {
        super.setUp();
        accountTransaction = new AccountTransaction();
     }

    void tearDown() {
        super.tearDown();
    }

    void testSomething() throws Exception {
        assertEquals("Expected",AccountTransaction.something())
    }
}
```

Test-First Tips

Use a Second LiveTemplate to generate the additional test methods and assertions.

```
public void test$method$() throws Exception {
    assertEquals("expected","actual")
}
```

Test-First Tips

Use “Quick Fix” to generate the actual code.

Eclipse:

- Ctrl-1

IntelliJ:

- Alt-Enter

They quickly create classes and/or methods that are mentioned in the test code but do not yet exist in the actual code.

Test-First Tips

Start with What you Know

Use TDD even if you cannot figure out how to directly test the "meat" of the logic with xUnit asserts.

e.g. code that generates an Adobe Acrobat file...

- At least, check for the existence of the created file,
- and that the size of the file is in an expected range.

Test-After Tips

Limit Morphing Vague Expectations

"Fake it till you make it," only in moderation.

Does the **`getFullName()`** method of a service return First-space-Last, or Last-comma-First? It doesn't matter? Then, just try it one way, and then change your expectation if you guessed wrong.

```
assert("John Smith",getFullName())
```

Error. Expected: John Smith, Actual: Smith, John

```
assert("Smith, John",getFullName())
```

Test-After Tips

But, be careful!

No clue what a method is supposed to return?
Do not take the actual for granted.

```
assert("X",getEncodedValue())
```

Error. Expected: X, Actual: hh3j3bkuy3edovuoyg98656q

||
V

```
assert("hh3j3bkuy3edovuoyg98656q",getEncodedValue())
```

All this proves is WAC (it “works as coded”)!

Test-After Tips

Hand-Code the Expectations

Every time!

```
Expected: <"Quoted Text">
  Actual: <”Quoted Text”>
```

Otherwise, you may not notice that you are actually condoning the wrong values: smart quotes vs. dumb quotes, letter l vs. number 1, missing ending period, etc.

How'd I do?

As a _____(role)_____

I want TDD in my tool box

So that _____(expectation)_____

http://www.ChiefSimplicityOfficer.com

If you would like to schedule this, or another of my presentations, for your group, don't hesitate to contact me.

craig@chiefsimplicityofficer.com
714-955-4025