An Introduction to
Behaviour Driven Testing
(BDT)

Mark Crowther – Empirical Pragmatic Tester
mark@cyreath.co.uk
About this slide pack.

This slide pack provides a brief overview of Behaviour Driven Testing.

Contact Mark Crowther to learn more about this approach, the tools and techniques used and with any queries or feedback you may have.

Copyright notice

This document is copyright of Mark Crowther - © Mark Crowther 2009. The content and trademarks are the property of and copyright of their respective owners. All rights reserved.

You may not, except with the express written permission of Mark Crowther make derivative works or commercially exploit the content or the publication. Nor may you transmit it or store it in any website or other form of electronic retrieval system except as permitted by this copyright notice.

Any redistribution or reproduction of part or all of the content or publication in any form is prohibited other than the following:

you may print or download the complete document or extracts of the content to a local hard disk and you may transmit it to an individual third party for personal and non-commercial use only; but only if you acknowledge Mark Crowther as the source of the publication and provide this copyright notice intact.

For more information contact Mark Crowther.
Q) What is Behaviour Driven Testing anyway?


http://www.scotlands-enchanting-kingdom.com/images/confused-monkey.jpg
Q) What is **Behaviour Driven Testing** anyway?

A) The testing perspective on **Behaviour Driven Development**
Behaviour Driven Testing

Gosh-dang it... Behavior Driven wut?

Ooee ugh... ee OO oo?
Test Driven Development

Behaviour Driven Development

Behaviour Driven Testing
Behaviour Driven Testing

Here comes the explanation

http://www.davisandco.com/products/yourattentionplease/

Copyright ©Mark Crowther 2009
Behaviour Driven Testing

November 2007 - QCon Developer Conference

- Dan North presents Behaviour Driven Development

- Dan first wrote a paper about his idea in 2003
Dan made the following observations:

In **TDD**: 

When a developer writes a **UNIT TEST**

they are thinking of **BEHAVIOUR** that **CODE**

not yet written - will **ENACT**

in the future
once the code is **WRITTEN**
Dan made the following observations:

Developers write **Unit Tests** before any code is written

At this point they’re **not** tests...

They can’t be – **there’s nothing to test** at this point

they’re **EXAMPLES** or **SPECIFICATIONS**

of **behaviour** future code will enact
Dan made the following observations:

- The code then gets **written**
- The Unit Tests then **pass**

NOW the Unit Test **will** become a test, a **regression test**

---

**Timeline**

- Requirement
- Unit Test
- Code
- Regression Test
Behaviour Driven Testing

In Behaviour Driven Development

- Requirement
  - Unit Test
  - Unit Test
  - Unit Test
  - Unit Test
 Behaviour Driven Testing

In Behaviour Driven Development

In BDD - Unit Test = Specification of Behaviour
that individual components of code should enact
Behaviour Driven Testing

In Behaviour Driven Development

As code is written to meet the ‘Specifications of behaviour’ for the component

**behaviour** starts to **emerge**

and the customer

**Requirements** start to **get delivered**
Behaviour Driven Testing

In Behaviour Driven Development

As further Requirements are delivered

Component Integration Testing

is possible and **usable functionality**

emerges allowing possible

deployment
Behaviour Driven Testing

Functionality that is expressed as BEHAVIOUR....

“Functional Testing is the system level testing of Characteristics the application exhibits and Behaviours it can enact”

Mark Crowther – A Testing Taxonomy
Why is focusing TESTING on BEHAVIOUR so important?
Why is a focus on BEHAVIOUR so important?

• **Users** usually don’t care about **technical implementation**
  they care about **BEHAVIOUR** of the software

The software was crap and never did what I wanted ...

... but I LOVE the way you implemented that CLASS!!

“...our clients don’t value the code as such; they value the things that the code does for them.” – Michael Bolton

Why is focusing TESTING on BEHAVIOUR so important?

• **User Acceptance Testing** is all about BEHAVIOUR the system can enact in response to user interaction.

• When we test at the **FUNCTIONAL** and **SYSTEM** level we test BEHAVIOUR.

• As testers we should NEVER be testing application CODE **directly**, that’s what DEVELOPERS are meant to do.

• Every bug we raise as testers should be described in terms of BEHAVIOUR.

**So how does this CHANGE what we do?**
Behaviour Driven Testing

As a test team following BDT we can now:

• focus our **TESTING** on system level **BEHAVIOUR** and stop thinking about testing **CODE**

• more easily **write tests up front** that are likely to **remain valid** as they’re not so tied to **implementation specifics**

• build closer relationships with the **Business Analysts** through the **Behaviour Clarification Reviews**

• help developers **deliver** more usable functionality **more quickly** as stories will be **qualified** increasing **development velocity**
As a test team following BDT we can now:

• write tests that are ‘Just Enough’ to cover the intended BEHAVIOUR of the system – so ensure good coverage

• meaning tests will all be a priority to run, all will be the ‘The Most Important Tests’
The Tester in a BDT environment needs to:

- Help with **STORY DEFINITION** to avoid **ILL DEFINED** requirements
- Focus on identifying **BUSINESS VALUE** and not **ABSTRACT IDEAS**
- Focus on **CUSTOMER USE** over the **TECHNICAL IMPLEMENTATION**
- Focus on **FINDING BUGS** and not on **CONFIRMING COMPLETENESS**

**More effort into TESTING the RIGHT things**
So how does this **CHANGE** what we do?

**Stakeholder / User** → **Business Analyst**

Stories / Requirements / Needs

**Business Analyst** ← **Tester**

Refined (Test) Scenarios / Automated Tests

**Tester** ← **Developer**

Code Deploy / Test Execution\(^0\) / Accepted system
So how does this CHANGE what we do?

**Exploratory Testing**
- Learn, plan and execute **at once**
- Look for **bugs**, **missing features** and **opportunities** for improvement

**Combination/Interaction Testing**
- Focus on **interactions between features**

**Scenario Testing**
- Use combined **real-world scenarios** that exercise **multiple** stories

**Business Cycle Testing**
- Execute scenarios based on end of day/month/qtr/year **business tasks**

**Non-Functional Testing**
- Performance, Security, Accessibility, Compatibility

Behaviour Driven Testing

So how does this CHANGE what we do?

The Test Team adopt an XP like approach – XP/SCRUM compatible
Components that enable BDT

- Manage **FEATURES** as **FUNCTIONAL REQUIREMENTS**
- Manage **SCENARIOS** as **TEST CASES**
- Use all **other modules** as standard to manage testing
Behaviour Driven Testing

Components that enable BDT

Test Scenario

Test Case

Feature: Addition
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers

Scenario: Add two numbers
Given I have entered 50 into the calculator
And I have entered 70 into the calculator
When I press add
Then the result should be 120 on the screen
Behaviour Driven Testing

Components that enable BDT

```javascript
browser.open @the_url
browser.window_focus
browser.window_maximize
browser.type("q","hello world")
browser.click("btnG")
browser.wait_for_page_to_load("5000")
browser.get_title.should == "hello world – Google Search"
```
Components that enable BDT

```ruby
# Sample script showing indexing into an array to create a string for search, data driven test, etc.

testData = %w{Software Testing Banana Monkey}
partTitle = " - Google Search"

- testData.each do |searchString|
  puts searchString
  puts searchString+partTitle
end
```


http://oreilly.com/catalog/9780596516178

Copyright ©Mark Crowther 2009
Behaviour Driven Testing

Components that enable BDT

RSpec Code Examples

```
# Should give a PASS if the browser title is 'hello world - Google Search'

expected: "banana monkey - Google Search",

got: "hello world - Google Search" (using ==)

# Should give a FAIL has title is hello world but we're looking for banana monkey

C:\Users\Mark Crowther\Documents\4. Training Services\Course 3 - Open Sour

browser.click("btnG")
browser.wait_for_page_to_load("5000")

driver.get_title.should == "Banana monkey - Google Search"
end # it
```

http://rspec.info/

Behaviour Driven Testing

An outline of a **BDT process**

**BUSINESS ANALYST** identifies the **FEATURES** the **CUSTOMER/USER** wants

Testers use Cucumber **FEATURES** as **TEST SCENARIOS**

Tester writes **TEST CASES** as behaviour driven Cucumber **SCENARIOS**

Developer writes code that make **TEST CASES** **PASS**

Cucumber Feature = Test Scenario
Cucumber Scenario = Test Case

Copyright ©Mark Crowther 2009
Behaviour Driven Testing

An outline of a BDT process

**TEST CASES** get **AUTOMATED** by the testers

↓

Code is **DEPLOYED** and Test execute their **AUTOMATED TESTS**

↓

**BUGS** get fixed, automated tests run as **REGRESSION TESTS**

↓

When done Customer / User **ACCEPTS** the software

(Acceptance & Quality Criteria met)
Final thought...
why Ruby?
Behaviour Driven Testing

**Example Test Scenario:**

- Enter a number of search strings
- Check the string
- Check the string returned in the title
Behaviour Driven Testing

**JavaScript:**

```javascript
/* Sample script showing indexing into an array to create a string for search, data driven test, etc */

var testData=new Array("Software", "Testing", "Banana", "Monkey");
var testLoop=0;
var partTitle = " - Google Search";
var searchString;
var concatTitle;

while (testLoop<4)
{
    searchString = (testData[testLoop]);
    concatTitle = (testData[testLoop] + partTitle);
    document.write(searchString + "<br />");
    document.write(concatTitle + "<br />");
    testLoop++;
}
```

Copyright ©Mark Crowther 2009
**Behaviour Driven Testing**

**JavaScript:**

```javascript
/* Sample script showing indexing into an array to create a string for search, data driven test, etc */

var testData = new Array("Software", "Testing", "Banana", "Monkey");
var testLoop = 0;
var partTitle = " - Google Search";
var searchString;
var concatTitle;

while (testLoop < 4) {
    searchString = testData[testLoop];
    concatTitle = testData[testLoop] + partTitle;
    document.write(searchString + "<br/>");
    document.write(concatTitle + "<br/>");
    testLoop++;
}
```

Copyright ©Mark Crowther 2009

http://www.oreilly.com/catalog/9780596101992
Behaviour Driven Testing

Ruby:

```ruby
# Sample script showing indexing into an array to create a string for search, data driven test, etc.

testData = %w{Software Testing Banana Monkey}
partTitle = " - Google Search"

- testData.each do |searchString|
  puts searchString
  puts searchString
end
```

I eat Jelly Babies for BREAKFAST!
(and then write Ruby tests)
Final thought... why Ruby?

- Testers don’t get time to code constantly, the chosen language must:
  - be easy to learn
  - flexible in use
  - powerful

- Ruby is a fully fledged, programmatic language
  - It can be used to test GUI
  - Extended to test mid-tier applications
  - Used to write test harnesses
  - and is growing in popularity
End of the Presentation

Mark Crowther – Empirical Pragmatic Tester
mark@cyreath.co.uk