PRACTICE

# Research :

* When and why BDD was defined

November 2009 in London, Dan North

To provide software development and management teams with shared tools and a shared process to collaborate on software development.

* The most important aspects of BDD

1. Communication between business and development and testers is extremely focused as a result of common language (ubiquitous language).

2. Business needs tie directly to the code that is written.

3. Developers know what test cases they should write to accommodate TDD.

4. All the underlying benefits of TDD become more easily realized.

5. Code is easier to maintain.

6. Code is self documenting.

7. Stories are easier to “groom” – breakdown, task and plan.

8. Teams can be more agile.

9. Developers can be trained / on-boarded easier.

10. There is more visibility into team progress and status.

* Why is useful and when could be applied.

BDD offers more precise guidance on organizing the conversation between developers, testers and domain experts

Notations originating in the BDD approach, in particular the given-when-then canvas, are closer to everyday language and have a shallower learning curve compared to those of tools such as Fit/FitNesse

Tools targeting a BDD approach generally afford the automatic generation of technical and end user documentation from BDD "specifications"

* Which type of teams could implement BDD

Any

# Structure of a feature – Give an example

It's a collection of business outcomes

# Structure of a user story – Give an example

It's the representation of a feature which defines the scope of the feature along with its acceptance criteria.

**Structure:**

Title (one line describing the story)

Narrative:

As a [role]

I want [feature]

So that [benefit]

Acceptance Criteria: (presented as Scenarios)

Scenario 1: Title

Given [context]

  And [some more context]...

When  [event]

Then  [outcome]

  And [another outcome]...

Scenario 2: ...

**Sample:**

Story: Account Holder withdraws cash

As an Account Holder

I want to withdraw cash from an ATM

So that I can get money when the bank is closed

# Structure of a scenario – Give an example

The scenario title should say what’s different, should be described in terms of Givens, Events and Outcomes

**Sample:**

Scenario 1: Account has sufficient funds

Given the account balance is \$100

 And the card is valid

 And the machine contains enough money

When the Account Holder requests \$20

Then the ATM should dispense \$20

 And the account balance should be \$80

 And the card should be returned

Scenario 2: Account has insufficient funds

Given the account balance is \$10

 And the card is valid

 And the machine contains enough money

When the Account Holder requests \$20

Then the ATM should not dispense any money

 And the ATM should say there are insufficient funds

 And the account balance should be \$20

 And the card should be returned

Scenario 3: Card has been disabled

Given the card is disabled

When the Account Holder requests \$20

Then the ATM should retain the card

And the ATM should say the card has been retained

Scenario 4: The ATM has insufficient funds

...

# Differences between BDD and BDT

This is a BDD scenario (from [Dan North](http://dannorth.net/introducing-bdd/), a man I respect and admire):

*+Scenario 1: Account is in credit+  
Given the account is in credit  
And the card is valid  
And the dispenser contains cash  
When the customer requests cash  
Then ensure the account is debited  
And ensure cash is dispensed  
And ensure the card is returned*

This is that BDD scenario turned into testing:

*+Scenario 1: Account is in credit+  
Given the account is in credit  
And the card is valid  
And the dispenser contains cash  
When the customer requests cash  
Then****check that****the account is debited  
And****check that****cash is dispensed  
And****check that****the card is returned****And check that nothing happens that shouldn’t happen and everything else happens that should happen for all variations of this scenario and all possible states of the ATM and all possible states of the customer’s account and all possible states of the rest of the database and all possible states of the system as a whole, and anything happening in the cloud that should not matter but might matter.***