

BDD FRAMEWORKS & TFS

Rob Maher



MIKE HENRY

DriveRight

Founder – Dan North

- 2006 ish
- <http://DanNorth.net>
- <http://dannorth.net/introducing-bdd>

BDD is a second-generation, outside-in, pull-based, multiple-stakeholder, multiple-scale, high-automation, agile methodology. It describes a cycle of interactions with well-defined outputs, resulting in the delivery of working, tested software that matters. – Dan North 2009



MIKE HENRY

DriveRight

Why?

- Communication
- Outside In
- TDD Problems

The screenshot displays a Drupal website interface with several highlighted error messages and a full node page. The top navigation bar includes 'My Account' and 'Log Out'. Below the site name, there are tabs for 'Primary', 'Links', 'Menu', 'Goes', and 'Here'. The main content area features a search bar, a monthly archive, a user login form, and a nested menu. The full node page shows a title 'My full node page', a text block, a shopping cart icon, and a comments section. The error messages are highlighted in colored boxes: a red box for a general error, a blue box for a help message, a green box for a status message, and a yellow box for a warning message. The comments section includes a form for adding a new comment and a list of existing comments.



MIKE HENRY

DriveRig

BDD – In the beginning

- Tests should output sentences

CustomerLookup

- finds customer by id
- fails for duplicate customers
- ...

BDD

- Test Methods start with “Should”



MIKE HENRY

DriveRight

BDD

- An expressive test name is helpful when a test fails

BDD

- “Behaviour” is a more useful word than “test”

BDD

- Determine the next most important behaviour

BDD

- Requirements are behaviour, too

BDD

- BDD provides a “ubiquitous language” for analysis
- Given When Then

*As a customer,
I want to withdraw cash from an ATM,
so that I don't have to wait in line at the bank.*

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*



MIKE HENRY

DriveRight

BDD

- Acceptance criteria should be executable

```
public class AccountsInCredit :Given {  
    public void setup(World world) { ... }  
}
```

```
public class CardsValid :Given {  
    public void setup(World world) {...}  
}
```

and one for the event:

```
public class CustomerRequestsCash :Event {  
    public void occurIn(World world) { ... }  
}
```

BDD

- Context Specification

*As a customer,
I want to withdraw cash from an ATM,
so that I don't have to wait in line at the bank.*

Scenario 1: Account is in credit

*When the customer requests cash from an account that is in credit
It Should ensure the account is debited*

Feature Injection

In order to <achieve some outcome which contributes to the vision>

As a <stakeholder>

I want <some other stakeholder>

<to do, use or be restricted by something>



MIKE HENRY

DriveRight

Tools

- JBehave -> RBehave -> RSpec -> Cucumber
- MSpec
- StoryQ
- SpecFlow
-

MSpec

- Uses Context Specification
- Lambdas (love or hate) - = () =>
- Can output a html report

StoryQ

- Uses Given When Then
- Very strict format required
- One way requirement translation
- Parser not runner



MIKE HENRY

DriveRight

SpecFlow

- VS Add in
- Feature files
- Regex
- Parser

