Behavior Driven Development (BDD)

With Mocking

An amazing set of agile tools that helps to create kick ass software

Eric Hosick (erichosick)
Throughout this lecture, please remember:

Agile doesn't fix problems in your company. It simply brings them to light quicker.

As such:

Don't be Quick to Blame Agile
Software Without Requirements???
Agile Manifesto: [Favor] working software over comprehensive documentation.

We need requirements. They describe the features/behavior of the program being developed.

If we need requirements, then why not write requirements in a way that they can be verified similar to source code?
Behavior Driven Development

What is it? The name gives it all away...

- **Behavior/Feature**
  - What your software will do described through requirements (mocking, stories, begging).

- **Driven**
  - To cause or guide the movement of something.

- **Development**
  - The creation of software through engineering and coding.

- **Behavior Driven Development**
  - The development of software guided directly by described behavior and features (and mocking).
Given, When, Then

- **Given**: Eric is authorized
- **When**: he visits the "About Me" Page
- **Then**: he should still be authorized and the name "Eric" is shown

*What the software will look like to the user*

*Things that the user will do.*

*What the user should expect.*
Discovering Behavior

- Mockup the system using mocking software (Balsamiq for example).
  - Create user stories from mockup and or
  - Create mockup from user stories
  - Hack away on Research and Development but keep that code out of production.
- Discover ALL behavior and features of the mockup and write Scenarios with Gherkin
- Only then should you write code being pushed to production.
Example Mockup

http://balsamiq.wpengine.netdna-cdn.com/images/samples/mytunez.png
Why It is So Cool

● **Aligns With Agile Manifesto:**
  ○ Favor code over documentation

● **Forces Product Owner To Focus**
  ○ Mock-out and verify what stakeholders want (Lean Development) before writing a line of code.

● **Forces Developer To Focus**
  ○ Developer codes exactly what is needed.

● **Requirements and code Are "Verified"**
  ○ Continuous integration of requirements
  ○ Continuous integration of code
Why It is So Cool - Part 2

● Describes Your System Twice
  ○ The behavior of your system is written in a programming language.
  ○ The behavior of your system is written in a DSL.
  ○ Makes your software more robust.

● Same Scenarios In Different Languages
  ○ Ruby (Cucumber), C# (SpecFlow), PHP (Behat), Java (???)

● Iterative Development
  ○ Implement scenarios and push to production as fast as possible.
Why It is So Cool - Part 3

- 100% Coverage of the Behavior of Your Software
  - Easy to refactor
  - Any bugs caused by changes are found out: if you changed the behavior of your program you will find out!
  - See Describes your system twice.

YOU CAN'T BREAK YOUR PROGRAM!
Arguments Against BDD

- **Code Isn't Dry**
  - BDD is too wordy. But! Depends on the language and depends on the robustness of your code.

- **Product Owner Can't Use BDD**
  - But - The Domain Specific Language can be written to a level appreciated by the Product Owner.
  - If they can't explain what they want to you in their own language, you need a new product owner.

- **Assumes you Know the Problem**
  - But - If you don't know the problem then how can you code a solution?
  - Doing R&D - Don't use BDD. Have fun and Hack.
  - Sticking that code in production? Not without BDD.
Arguments Against BDD Part 2

● **Don't Use In Startup - No Upfront Value**
  ○ *But!* BDD forces the startup to focus on their core products and work closely with customers.
  ○ Remove code before it is even written!

● **Product Owner Can's Use BDD**
  ○ *But - The Domain Specific Language can be written to a level appreciated by the Product Owner.*
  ○ If they can't explain what they want to you in their own language, you need a new product owner.

● **Assumes You Know the Problem**
  ○ *But - If you don't know the problem then how can you code a solution?*
  ○ *Doing R&D? - Don't use BDD. Have fun and Hack.*
  ○ *Sticking code in production? Not without BDD.*
Arguments Against BDD Part 3

• *Have to Change the BDD A Lot*
  ○ *If you are changing the BDD a lot does that mean BDD is broken? OR*
  ○ *Does that mean the Product Owner had you write something they really didn't want?*
Questions?

source: http://www.flickr.com/photos/katemonkey/122489910/