Building in Quality: The Beauty of Behavior Driven Development (BDD)

Larry Apke - Agile Coach

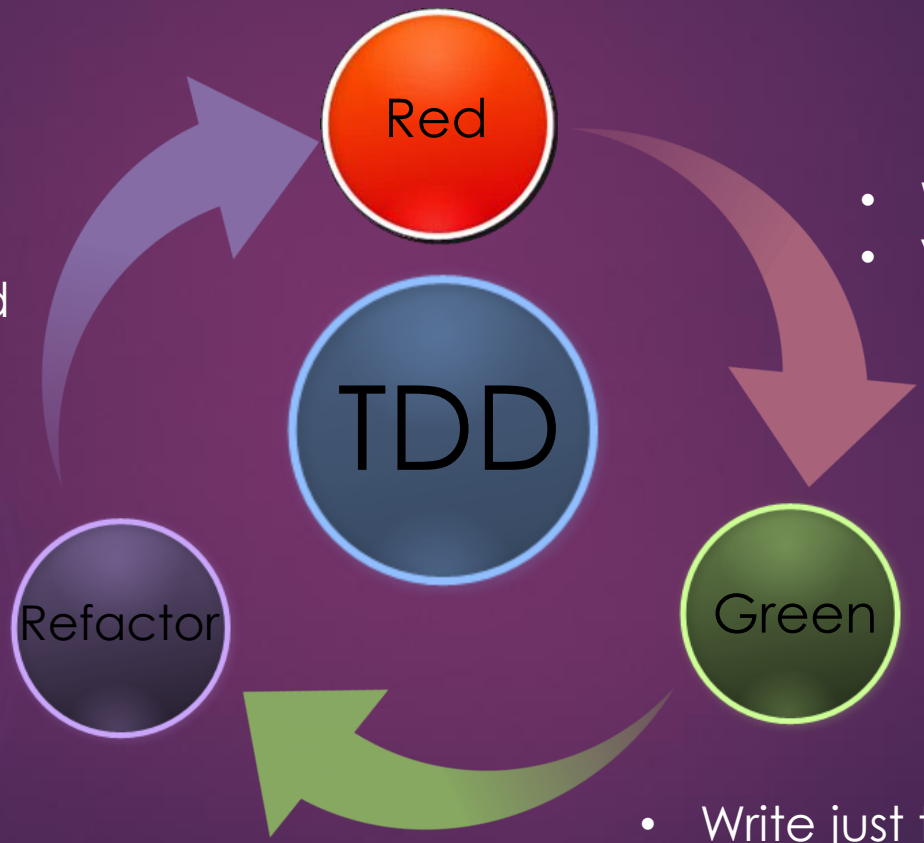
Deming on Quality

"Quality comes not from inspection, but from improvement of the production process."

"We cannot rely on mass inspection to improve quality… As Harold S. Dodge said many years ago, 'You cannot inspect quality into a product.' The quality is there or it isn't by the time it's inspected."

BDD Begins with Test Driven Development (TDD)

- Refactor the code
- Coherence is ensured by the unit tests

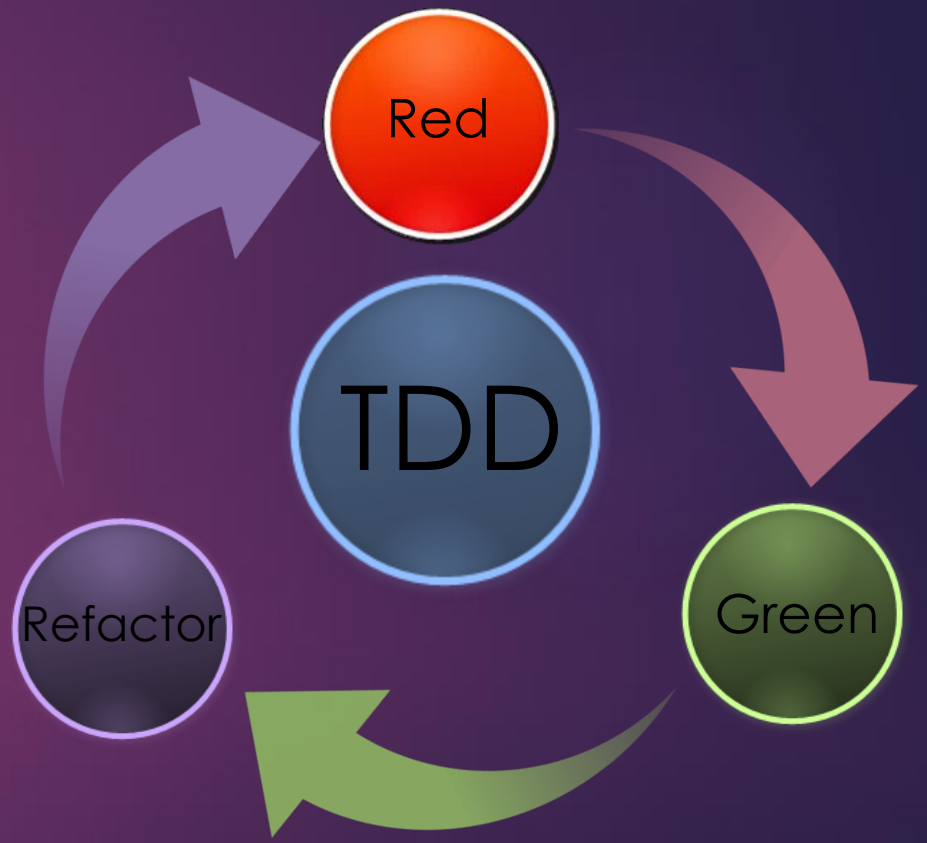


- Write a unit test
- Validate that the test fails

- Write just the minimum code necessary
- Validate that the test passes successfully

Why Do TDD?

- "Ilities" – Maintainability, Flexibility, Extensibility
- High Percentage of Test Code Coverage
- Streamlined Codebase
- Cleaner Interfaces
- Easier Refactoring of Code
- Tests Provide Some Code Documentation
- Bottom Line - Higher Quality



An Expert Speaks on TDD

"It is a myth that we can get systems "right the first time." Instead, we should implement only today's stories, then refactor and expand the system to implement new stories tomorrow. This is the essence of iterative and incremental agility. Test-driven development, refactoring, and the clean code they produce make this work at the code level."

An Expert Speaks on TDD

"If the discipline of requirements specification has taught us anything, it is that well-specified requirements are as formal as code and can act as executable tests of that code!"

An Expert Speaks on TDD

"So if you want to go fast, if you want to get done quickly, if you want your code to be easy to write, make it easy to read."

An Expert Speaks on TDD

"Indeed, the ratio of time spent reading versus writing is well over 10 to 1. We are constantly reading old code as part of the effort to write new code. ...[Therefore,] making it easy to read makes it easier to write."

An Expert Speaks on Being a Professional Developer

"Perhaps you thought that "getting it working" was the first order of business for a professional developer…The functionality that you create today has a good chance of changing in the next release, but the readability of your code will have a profound effect on all the changes that will ever be made."

An Expert Speaks on Being a Professional Developer

"Programmers who satisfy themselves with merely working code are behaving unprofessionally."

An Expert Speaks on Being a Professional Developer

"Writing clean code is what you must do in order to call yourself a professional. There is no reasonable excuse for doing anything less than your best."

Being a Professional Developer

In addition to Test Driven Development and writing testable, readable code, there are a number of other things that go into making one a professional programmer.

Your code must be secure

Your code must be performant

Your code must be deployable

Your code must integrate

In other words, code quality must be injected into the code by the professional programmer. Quality cannot be inspected into the code ex post facto (after the fact).

So, Why Doesn't Everyone Do TDD?

- Most developers have not learned TDD and there is a learning curve
- TDD is extremely disciplined
- There is a perception that development is slower
- It is difficult to introduce with legacy code
- Developers often need to mock objects and learn mocking

"I kept coming across the same confusion and misunderstandings. Programmers wanted to know where to start, what to test and what not to test, how much to test in one go, what to call their tests, and how to understand why a test fails." - Dan North - Introduction to BDD

Along Comes BDD

"My response is behaviour-driven development (BDD). It has evolved out of established agile practices ... Over time, BDD has grown to encompass the wider picture of agile analysis and automated acceptance testing."

Created a "ubiquitous language for the analysis process itself!"

Over time they found that the conversations between business and development where richer.

Why Do BDD?

TDD builds it right, BDD builds the right thing.

While there is no silver bullet to software development, BDD (when coupled with continuous integration) comes as close as anything I have found.

My blog - "10 Reasons Why BDD Changes Everything" was published on 3/6/2012 and still is the most visited page on my website.

Ten Reasons Why BDD Changes Everything

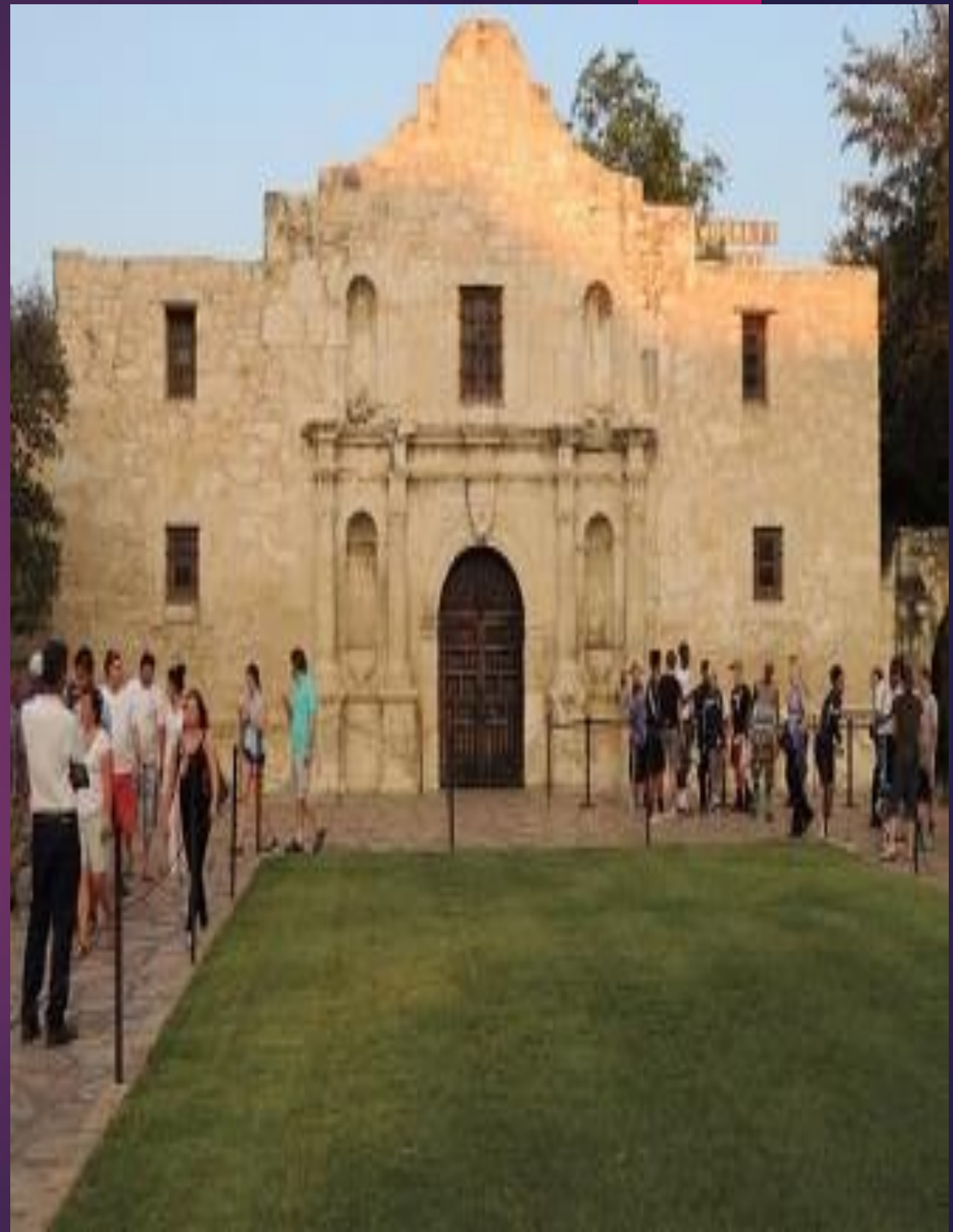
- Communication between business and development is extremely focused as a result of common language.
- Business needs tie directly to the code that is written.
- Developers know what test cases they should write to accommodate TDD.
- All the underlying benefits of TDD become more easily realized.
- Code is easier to maintain.
- Code is self documenting.
- Stories are easier to “groom” – breakdown, task and plan.
- Teams can be more agile.
- Developers can be trained / on-boarded easier.
- There is more visibility into team progress and status.

BDD – Real World Stories

Expressing Acceptance Criteria as BDD scenarios breaks the bottle neck in story writing for a small financial services company in Scottsdale, Arizona.

BDD – Real World Stories

BDD provides support necessary to successfully deliver MVP into production in record time for financial services company in San Antonio, Texas.



Adopting BDD – Three Simple Steps

Step One

Write our story acceptance criteria using BDD scenarios

Step Two

Leverage these BDD scenarios to write our automated tests first in true TDD fashion

Step Three

Leverage a BDD Tool like Cucumber to map human-readable BDD scenarios to the automated tests

Step One – Write Acceptance Criteria in BDD Style

Step One is accomplished by the three amigos - Product Owner, Developer(s) and Quality Assurance Person(s).

Step One is accomplished during Story Refinement which is where we go into detail for the stories that we believe we will take during the next iteration.

This is the first step in transitioning to BDD and the one that brings the most benefit. It was a pleasant surprise to Dan North and crew, something that they did not plan. Better communication equals better quality.

Step One – Write Acceptance Criteria in BDD Style

The Story

Customer Withdraws Cash

As a customer,

I want to withdraw cash from an ATM,

So that I don't have to wait in line at the bank.

Step One – Write Acceptance Criteria in BDD Style

The Story

Customer Withdraws Cash

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

Step One – Write Acceptance Criteria in BDD Style

The Story

Customer Withdraws Cash

Scenario 2 - Account is overdrawn past the overdraft limit

Given the account is overdrawn

And the card is valid

When the customer requests cash

Then ensure a rejection message is displayed

And ensure cash is not dispensed

And ensure the card is returned

Step Two – Use BDD Scenarios to Write Tests

In step two we leverage BDD to write our tests and begin the process of TDD. BDD scenarios can be written directly into tests using comments.

Step Two – Use BDD Scenarios to Write Tests

```
public class AccountIsInCreditTest extends TestCase {
@Setup
//Scenario 1 - Account is In Credit
// Given the account is in credit
//  And the card is valid
//  And the dispenser contains cash
//Write code here to mock out account and add credit, mock out card and set as valid, etc.
```

Step Two – Use BDD Scenarios to Write Tests

```
//When the customer requests cash
Call CustomerRequestsCash (Amount) //call method on class and pass values
@Test
//Then ensure the account is debited
//  And ensure cash is dispensed
//  And ensure the card is returned
Assert.AssertEquals(ExpectedAccountBalance, Account.Balance);
Assert.AssertEquals(CashDispensed, True);
Assert.AssertEquals(CardReturned, True);
}
```

Step Three – Use a BDD Tool

- BDD Tool will take human-readable scenarios and use regular expressions to map the human readable scenarios to Unit Tests.
- BDD Tools will create a unit test and shell automatically (referred to as a step definition file) that helps in the creation of test.
- When the BDD Tool runs the tests, the output shows which tests passed and failed against the human-readable scenarios.
- There are BDD Tools for all major languages. I recommend Cucumber for Java and Ruby.

Step Three – Use a BDD Tool

Scenario: Valid Log On

Given user navigates to logon page

When they enter valid user name and password

Then login should be successful

Step Three – Use a BDD Tool

```
@Given("^user navigates to logonpage$")
  pending # express the regexp above with the code you wish you had
end

@When("^they enter valid Username as \"([^\"]*)\" and Password as \"([^\"]*)\"$")
  pending # express the regexp above with the code you wish you had
end

@Then("^login should be successful$")
  pending # express the regexp above with the code you wish you had
end
```

Step Three – Use a BDD Tool

```
@Given("^user navigates to logonpage$")
    public void goToLogonPage() {
    driver = new FirefoxDriver();
    driver.navigate().to(“logonpage.htm");
  } end

@When("^they enter valid Username as \"([^\"]*)\" and Password as \"([^\"]*)\"$")
  public void ValidUserNameAndPassword(String arg1, String arg2) {
    driver.findElement(By.id("email")).sendKeys(arg1);
    driver.findElement(By.id("pass")).sendKeys(arg2);
    driver.findElement(By.id(“submit")).click();
  }
end
```

Step Three – Use a BDD Tool

```
#encoding: utf-8
Feature: Logging On

  Scenario: Valid Log On          # /logom:8
    Given user navigates to logon page # /step_defs/logon.rb:3
    When they enter valid user name and password #
/step_defs/logon.rb:7
    Then login should be successful          # /step_defs/logon.rb:14

1 scenario (1 passed)
3 steps (3 passed)
0m0.005s
```

Cucumber

Cucumber Features

1 scenario (1 failed)
8 steps (1 failed, 2 skipped, 5 passed)
Finished in 0m1.105s seconds

Feature: limit the amount of products displayed on a page

In order to reduce strain on the site and overwhelm users
As an owner
I want to allow users to navigate through pages of products

Background:

- Given the products per page setting is set to **20**
- And there is a **category** with **30 products**
- When I view the **category**

Scenario: navigating pages

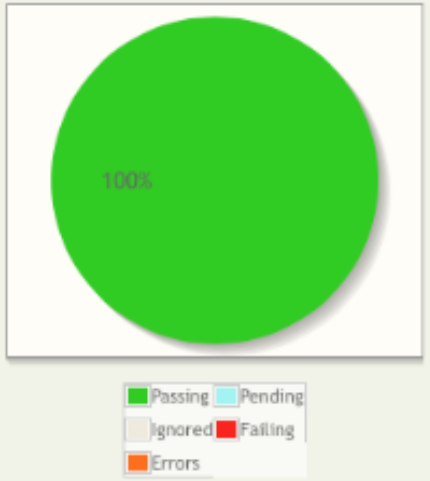
- Then I should see **20 products**
- When I click product page **2**
- Then I should see **5 products**

```
Expected at least 5 elements matching ".product", found 10.
<false> is not true.

/Users/robaldred/Sites/mvor/features/step_definitions/general.rb:114
/Users/robaldred/Sites/mvor/features/webrat/products/pagination.feature:25 :in `Then I should see 5 products'
```

```
112  end
113
114  Then /^I should see (\d+|some) ([\w|_]+)$/ do |count ,object|
115    if count == 'some'
116      response.body.should have_tag(".#{object.singularize}",:minimum => 3)
```

- When I click product page **1**
- Then I should see **20 products**



Requirements Overview

Requirement Type	Total	Pass	Fail	Pending	Ignored	Untested
Capabilities	2	2	0	0	0	0
Features	2	2	0	0	0	0
Stories	2	2	0	0	0	0
Acceptance Criteria (tests)	3	3	0	0	0	

Test Result Summary

Test Type	Total	Pass	Fail	Pending	Ignored
Automated	3	3 (100%)	0 (0%)	0 (0%)	0 (0%)
Manual	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Total	3	3 (100%)	0 (0%)	0 (0%)	0 (0%)

Capabilities (2)

Show 25 entries Search:

ID	Capability	Features	Auto. Tests					Manual Tests					Coverage
	Jvm types	1	*1*	1	0	0	0	*0*	0	0	0	0	100%
	Thread safety	1	*2*	2	0	0	0	*0*	0	0	0	0	100%

Showing 1 to 2 of 2 entries Previous 1 Next

Acceptance Tests (3)

Show 25 entries Search:

Acceptance Tests	Steps	Stable	Duration (seconds)
Running multiple factories should lead to different results	7		0.05
Running one factory should return a basic POJO filled with basic types	8		0.4
Running one factory should return a filled in POJO	8		0.07

Showing 1 to 3 of 3 entries Previous 1 Next

Helpful Links

Dan North - Introduction to BDD

The Cucumber Book: Behaviour-Driven Development for Testers and Developers

BDD in Action: Behavior-driven development for the whole software lifecycle

Specification by Example: How Successful Teams Deliver the Right Software

Cucumber Tool

Serenity Tool

Clean Code: A Handbook of Software Craftsmanship

Questions