What are the primary differences between TDD and BDD? [closed]

Asked 16 years, 4 months ago Modified 3 years, 1 month ago

Viewed 45k times

136

Test Driven Development has been the rage in the .NET community for the last few years. Recently, I have heard grumblings in the ALT.NET community about BDD. What is it? What makes it different from TDD?

unit-testing tdd bdd

Share

Improve this question

Follow

edited Apr 30, 2012 at 9:01

user257111

asked Aug 5, 2008 at 15:58

NotMyself
30k ● 17 ● 57 ● 74

2 See also, programmers.stackexchange.com/q/135218/76176. This question is more on topic there. – Evan Carroll Dec 2, 2013 at 5:18

TDD is for micro tests. BDD is for requirements or macro tests. Listen to episodes 1 through 8 about the Test Pyramid and it will explain these levels: agilenoir.biz/series/agile-thoughts – Lance Kind Apr 4, 2019 at 5:42

12 Answers

Sorted by: Highest score (default)

108

I understand BDD to be more about **specification** than **testing**. It is linked to Domain Driven Design (don't you love these *DD acronyms?).

It is linked with a certain way to write user stories, including high-level tests. An example by Tom ten Thij:

```
Story: User logging in
  As a user
  I want to login with my details
  So that I can get access to the site

Scenario: User uses wrong password

  Given a username 'jdoe'
  And a password 'letmein'

  When the user logs in with username and password
```

```
Then the login form should be shown again
```

(In his article, Tom goes on to directly execute this test specification in Ruby.)

The pope of BDD is Dan North. You'll find a great introduction in his Introducing BDD article.

You will find a comparison of BDD and TDD in this video. Also an opinion about BDD as "TDD done right" by Jeremy D. Miller

March 25, 2013 update

The video above has been missing for a while. Here is a recent one by Llewellyn Falco, BDD vs TDD (explained). I find his explanation clear and to the point.

Share Improve this answer Follow

edited Aug 7, 2015 at 13:04

answered Aug 5, 2008 at 16:36

Christian Lescuyer
19.2k ● 6 ● 50 ● 45

10 The video link seems to have gone bad – James Nail May 19, 2011 at 16:01

1 Christian, what was the video title and speaker name? so we can track it down – smci Nov 15, 2012 at 6:46

1 Above 'Tom Ten Thij' link is dead by now.. here's live @ - tomtenthij.nl/2008/1/25/… – Kundan Pandit Jan 19, 2015 at

6:41

Here is a short game that teaches the main points of BDD: agilenoir.biz/en/am-i-behavioral-or-not – Lance Kind Apr 5, 2019 at 3:09

16

To me primary difference between BDD and TDD is focus and wording. And words are important for communicating your intent.

TDD directs focus on testing. And since in "old waterfall world" tests come after implementation, then this mindset leads to wrong understanding and behaviour.

BDD directs focus on behaviour and specification, and so waterfall minds are distracted. So BDD is more easily understood as design practice and not as testing practice.

Share Improve this answer Follow

answered Sep 8, 2008 at 18:36

Juha Pohjalainen
475 ● 5 ● 11

3 TDD has nothing to do with the "waterfall" software design lifecycle. If anything, TDD is SDLC agnostic. The intent of TDD is to write the minimum amount of code required to get a test to pass. In a way the test becomes the technical specification for the code to adhere to. – Gavin Baumanis Mar 14, 2016 at 7:57

2 TDD is "Test first" and can work quite well with Agile. This isn't accurate. – Terrance Aug 29, 2017 at 20:08

13

There seem to be two types of BDD.

The first is the original style that Dan North discusses and which caused the creation of the xBehave style frameworks. To me this style is primarily applicable for acceptance testing or specifications against domain objects.

The second style is what Dave Astels popularised and which, to me, is a new form of TDD which has some serious benefits. It focuses on behavior rather than testing and also small test classes, trying to get to the point where you basically have one line per specification (test) method. This style suits all levels of testing and can be done using any existing unit testing framework though newer frameworks (xSpec style) help focus one the behavior rather than testing.

There is also a BDD group which you might find useful:

http://groups.google.com/group/behaviordrivendevelopment/

Share Improve this answer Follow

answered Sep 10, 2008 at 16:00

Colin Jack

8

Test-Driven Development is a test-first software development methodology, which means that it requires writing test code before writing the actual code that will be tested. In Kent Beck’s words:

> The style here is to write a few lines of code, then a test that should run, or even better, to write a test that won't run, then write the code that will make it run.

After figuring out how to write one small piece of code, now, instead of just coding on, we want to get immediate feedback and practice "code a little, test a little, code a little, test a little." So we immediately write a test for it.

So TDD is a low-level, technical methodology that programmers use to produce clean code that works.

Behaviour-Driven Development is a methodology that was created based on TDD, but evolved into a process that doesn’t concern only programmers and testers, but instead deals with the entire team and all important stakeholders, technical and non-technical. BDD started out of a few simple questions that TDD doesn’t answer well: how much tests should I write? What should I actually test—and what shouldn’t I? Which of the tests I write will be in fact important to the business or to the overall quality of the product, and which are just my over-engineering?

As you can see, such questions require collaboration between technology and business. Business stakeholders and domain experts often can tell engineers what kind of tests sound like they would be useful—but only if the tests are high-level tests that deal with important business aspects. BDD calls such business-like tests “examples,” as in “tell me an example of how this feature should behave correctly,” and reserves the word “test” for low-level, technical checks such as data validation or testing API integrations. The important part is that while *tests* can only be created by programmers and testers, *examples* can be collected and analysed by the entire delivery team—by designers, analysts, and so on.

In a sentence, one of the best definitions of BDD I have found so far is that BDD is about “having conversations with domain experts and using examples to gain a shared understanding of the desired behaviour and discover unknowns.” The discovery part is very important. As the delivery team collects more examples, they start to understand the business domain more and more and thus they reduce their uncertainty about some aspects of the product they have to deal with. As uncertainty decreases, creativity and autonomy of the delivery team increase. For instance, they can now start suggesting their own examples that the business users didn’t think were possible because of their lack of tech expertise.

Now, having conversations with the business and domain experts sounds great, but we all know how that often ends up in practice. I started my journey with tech as a programmer. As programmers, we are taught to *write code*—algorithms, design patterns, abstractions. Or, if you are a designer, you are taught to *design*—organize information and create beautiful interfaces. But when we get our entry-level jobs, our employers expect us to "deliver value to the clients." And among those clients can be, for example... a bank. But I could know next to nothing about banking—except how to efficiently decrease my account balance. So I would have to somehow translate what is expected of me into code... I would have to build a bridge between banking and my technical expertise if I want to deliver any value. BDD helps me build such a bridge on a stable foundation of fluid communication between the delivery team and the domain experts.

Learn more

If you want to read more about BDD, I wrote a book on the subject. **“Writing Great Specifications”** explores the art of analysing requirements and will help you learn how to build a great BDD process and use examples as a core part of that process. The book talks about the ubiquitous language, collecting examples, and creating so-called executable specifications (automated tests) out of the examples—techniques that help BDD teams deliver great software on time and on budget.

If you are interested in buying “Writing Great Specifications,” **you can save 39%** with the promo code **39nicieja2** :)

Share Improve this answer Follow

edited Oct 25, 2021 at 11:59

Rizwan
103 ● 4 ● 24

answered Feb 12, 2017 at 16:43

thion
101 ● 1 ● 1

6

I have experimented a little with the BDD approach and my premature conclusion is that BDD is well suited to use case implementation, but not on the underlying details. TDD still rock on that level.

BDD is also used as a communication tool. The goal is to write executable specifications which can be understood by the domain experts.

Share Improve this answer Follow

answered Aug 27, 2008 at 20:59

Thomas Eyde
3,944 ● 3 ● 26 ● 33

2

With my latest knowledge in BDD when compared to TDD, BDD focuses on specifying what will happen next, whereas TDD focuses on setting up a set of conditions and then looking at the output.

Share Improve this answer Follow

edited May 21, 2012 at 18:26

yoozer8
7,489 ● 7 ● 62 ● 96

answered May 25, 2009 at 4:09

Uma Mahesh Varma

2

Behaviour Driven Development seems to focus more on the interaction and communication between Developers and also between Developers and testers.

The Wikipedia Article has an explanation:

[Behavior-driven development](#)

Not practicing BDD myself though.

Share Improve this answer Follow

edited Jan 27, 2013 at 12:29
user1873471

answered Aug 5, 2008 at 16:06
Michael Stum
181k ● 119 ● 407 ● 541

2

Consider the primary benefit of TDD to be design. It should be called Test Driven Design. BDD is a subset of TDD, call it Behaviour Driven Design.

Now consider a popular implementation of TDD - Unit Testing. The Units in Unit Testing are typically one bit of logic that is the smallest unit of work you can make.

When you put those Units together in a functional way to describe the desired Behaviour to the machines, you need to understand the Behaviour you are describing to the machine. Behaviour Driven Design focuses on verifying the implementers' understanding of the Use

Cases/Requirements/Whatever and verifies the implementation of each feature. BDD and TDD in general serves the important purpose of informing design and the second purpose of verifying the correctness of the implementation especially when it changes. BDD done right involves biz and dev (and qa), whereas Unit Testing (possibly incorrectly viewed as TDD rather than one type of TDD) is typically done in the dev silo.

I would add that BDD tests serve as living requirements.

Share Improve this answer Follow

edited Oct 25, 2021 at 11:59
Rizwan
103 ● 4 ● 24

answered May 28, 2015 at 22:36
phil v
191 ● 1 ● 4

2

It seems to me that BDD is a broader scope. It almost implies TDD is used, that BDD is the encompassing methodology that gathers the information and requirements for using, among other things, TDD practices to ensure rapid feedback.

Share Improve this answer

Follow

edited Oct 25, 2021 at 11:59

Rizwan

103 ● 4 ● 24

answered Aug 5, 2008 at 16:11

palehorse

27.4k ● 4 ● 42 ● 49

1

In short there is major difference between TDD and BDD In TDD we are majorly focused on Test data In BDD our main focus is on behavior of the project so that any non - programming person can understand the line of code on the behalf of the title of that method

Share Improve this answer

Follow

answered Aug 24, 2019 at 14:53

saurabh singh

21 ● 2

1

There is no difference between TDD and BDD. except you can read your tests better, and you can use them as requirements. If you write your requirements with the same words as you write BDD tests then you can come from your client with some of your tests defined ready to write code.

Share Improve this answer

Follow

edited Oct 25, 2021 at 12:00

Rizwan

103 ● 4 ● 24

answered Oct 7, 2014 at 8:52

Mihai
518 ● 4 ● 12

0

Here's the quick snapshot:

> - TDD is just the process of testing code before writing it!
> - DDD is the process of being informed about the Domain before each cycle of touching code!
> - BDD is an implementation of TDD which brings in some aspects of DDD!

Share Improve this answer Follow

edited Aug 24, 2019 at 15:05
Dharman ♦
33.2k ● 27 ● 99 ● 146

answered Jan 18, 2016 at 3:01
Snehal Masne
3,419 ● 3 ● 34 ● 52