Application Testing

Asked 16 years, 3 months ago Modified 7 years, 6 months ago

Viewed 212 times

3

Is the real benefit in `TDD` the actual testing of the application, or the benefits that writing a testable application brings to the table? I ask because I feel too often the conversation revolves so much around testing, and not the total benefits package.

tdd

Share

Improve this question

Follow

edited Jun 9, 2017 at 10:30

gprathour

15.3k ● 5 ● 71 ● 92

asked Sep 8, 2008 at 23:31

RobZ

330 ● 3 ● 6

6 Answers

Sorted by: Highest score (default)

7

TDD helps you design your software. The tests becomes the design. By writing the test first you think about your code from a consumer perspective, making a more user friendly and more compact software design.

Also, by applying TDD you typically end up writing your code in a way where you can supply test mocks and stubs. This leads to less coupled software, making it easier to change and maintain over time.

So I guess allot of the talk around TDD is about testing, but by doing that other big benefits follow, such as quality (coverage), flexibility (decoupling), better design (think as the consumer of the API).

Share Improve this answer

Follow

answered Sep 8, 2008 at 23:36

Jonas Follesø

6,531 ● 7 ● 42 ● 54

4

The real improvement is that it is a good way to force you to really think through the design and implementation. Then, once you've prepared the tests and written the code, solutions to unforeseen problems appear more easily.

Something that usually happens to me that is a good analogy: When I'm going to post a question to a forum or IRC channel, I like to have the problems well written and fully described, many times the process of preparing a well written and complete description of the problem magically makes the solution appear.

Share Improve this answer

Follow

answered Sep 8, 2008 at 23:37

Vinko Vrsalovic

340k ● 55 ● 340 ● 373

I often have the same thing happen, or I find I can't actually define the problem and therefore have to go away and thing about it a while longer... ultimately splitting the issue into smaller ones and in doing so finding the solutions for each. Its a good habit to get into: "talking" probs through.
– Phillip B Oldham Nov 11, 2008 at 13:02

1

The real benefit of TDD is *supposed* to be that it allows you to modify/refactor/enhance your application without worrying about whether you've broken existing functionality. The fact that writing unit tests tends to result in loosely coupled code and better architecture isn't necessarily the *point* of TDD, but I think it's hard to have one without the other.

You can't really experience the benefit of TDD unless you have unit tests with good coverage. In order to do that, you're going to have to write testable code. That's why the two are often used in conjunction or in place of one another.

Share Improve this answer

Follow

answered Sep 8, 2008 at 23:36

Kevin Pang

41.4k ● 38 ● 122 ● 173

0

Automated testing is such a time saver and confidence booster when you are developing a product that you'll ship multiple versions of. With automated tests, you know that you haven't broken anything between versions. This

especially helpful when your product is something that people can write add-ons for - you don't want to break their add-ons between versions.

With TDD, you get a good suite of tests as you develop. Without TDD writing those tests is much more difficult.

Share Improve this answer

Follow

answered Sep 9, 2008 at 1:44

dan gibson

3,655 ● 3 ● 42 ● 57

0

Michael Feathers has an insightful blog post about this titled The Flawed Theory Behind Unit Testing. Seriously, go read it. The punch line is

> All of these techniques have been shown to increase quality. And, if we look closely we can see why: all of them force us to reflect on our code.

but you should read the full post for the context.

Share Improve this answer

Follow

answered Sep 12, 2008 at 15:26

Pat Notz

214k ● 31 ● 94 ● 92

-1

Automated testing keeps humans from doing a machine's job.

Test-driven development maximizes the amount of automated testing.

Beyond a certain point, of course, a human is still required. You reach diminishing returns when you try to apply TDD beyond that point.

Share Improve this answer

Follow

answered Sep 8, 2008 at 23:35

harpo

43.1k ● 14 ● 100 ● 133