Are you really using unit tests? [closed]

Asked 16 years, 1 month ago Modified 11 years, 11 months ago

Viewed 7k times

51

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 11 years ago.

I have been involved in a lot of projects, both old and new, and one thing that they have in common is that almost none of them have been using unit testing. I prefer to use it, but often the customer isn´t ready to pay for that, and suppose that the code just works as it should.

So, do you use unit testing in your projects, or do you rely on your coding skills?

unit-testing testing

Share

edited Nov 18, 2008 at 13:11

Improve this question

Follow

Ken
78.8k ● 32 ● 87 ● 101

asked Nov 13, 2008 at 9:08

Mikael Söderström
1,008 ● 8 ● 15

See is-unit-testing-worth-the-effort? – nawfal Jul 23, 2014 at 15:24

19 Answers

Sorted by: Highest score (default)

71

Using unit-testing *is* a coding skill.

I find it adds very little overhead to coding time. On top of that the code produced tends to be much easier to understand and to refactor, with an untold reduction in maintenance time.

A full discussion of the benefits here: Is Unit Testing worth the effort?

Share Improve this answer

Follow

edited May 23, 2017 at 12:17

Community Bot
1 ● 1

answered Nov 13, 2008 at 9:14

Ken
78.8k ● 32 ● 87 ● 101

1 The TDD mentioned on the discussion is one key point (to me). One doesn't have to first write the code and then the

tests separately but rather write the tests as the part of the developing process – Touko Nov 13, 2008 at 10:47

Good point. The problem is that when I work with others, they often have no idea about what testing is, they simply think of testing like "open the browser and test run the function. Does it work for me? Great". Btw, StackOverflow is not especially tested (see video form PDC about MVC). ;-)
– Mikael Söderström Nov 13, 2008 at 12:46

When you use MongoDB database in the your application, I feel Unit Testing would even be more useful especially in testing the CRUD operations of your Entities because MongoDB leaves it to the application developer to ensure Foreign Key Referential integrity is upheld. Unlike RDMS databases, MongoDB and a lot of other NoSQL databases leave enforcement of valid associations of data to the application developer. – CS Lewis Nov 27, 2015 at 5:44

38

I'll be honest. I've only recently started writing unit tests, and when I go back to modify an old DLL that's from my bad old days, I'll hunker down and write unit tests to get it near 100% code coverage. I say "near" because the last few percent can be hard to get to due to the way that the code is structured, that it didn't have mocking or unit testing in mind (this happens a lot with the abstract classes or classes that P/Invoke into native code). And while I understand that 100% code coverage is not the same thing as 100% of all code execution paths, I've found that having the tests there is a good way to tell when I'm doing something that's going to break a lot of things.

To be honest, this is probably one reason why it has taken many developers (including myself) so long to get around to adopting unit testing (let's not get into TDD just yet).

I'm not saying that any of these are legitimate reasons, but they did more or less go through my head, and I bet they went through some of yours too:

- First, there's a thought along the lines of "Oh, I've written mountains of code with zero tests, and seeing as I'm already crushed by that mountain, I don't possibly have the time to add several more mountains of test code on top of it."
- Second, nobody likes to talk about the fact that the classic code-run-debug cycle actually works "good enough" when you're
    - writing new code that does not alter the behavior of old code
    - working on a relatively small software project or a throwaway utility
    - the sole developer on a project (you can keep most of it in your head, up to a point)
- Third, unit tests are easier to appreciate when you're maintaining existing code, and if you're always writing new code, well, the benefits aren't immediately obvious. When your job is to churn out a utility and never touch it again, a developer can see little value in the unit test because they probably

won't be working on that utility or at the company by the time a maintenance programmer (who wishes there was a unit test) comes around.

- Fourth, unit testing is a fairly recent innovation. (Oh, hush ... I know the idea has been around forever, especially in mission-critical and DoD applications, but for the "Joe the ~~Plumber~~ Developer" types like me? Unit testing wasn't mentioned at all during my entire CS undergraduate career in the early part of this decade; in 2008, I hear it's a requirement for all projects from level 101 up. Visual Studio didn't even get a built-in testing framework until this year, and even then only in the professional edition.) Was it blissful ignorance on my part? Sure, and I think a lot of people who code because it's their day job (which is fine) simply aren't aware. If they are, then they know that they have to stay current, but when it comes to learning a new methodology (particularly one that involves writing more code and taking more up-front time when you needed that new feature done *yesterday*) means that it'll get pushed back. This is the long tail, and in a decade or so our talks about unit testing will seem as quaint as our mutterings about object-oriented programming enabling a new era of development during the 1990s. Heck, if *I've* started unit testing, the end must be near, because I'm usually an idiot.
- Fifth, unit testing some areas of code is really difficult, and this scared me away for a while. Multi-threaded event queues, graphical rendering, and

GUIs come to mind. Many developers realize this and think "well heck, unit testing would be great for library code but when was the last time I wrote just a class library." It takes a while to come around. To that end, no unit tests does not necessarily mean bad code lies ahead.

- Finally, the level of evangelism that sometimes comes from unit testing advocates can be intimidating and off-putting. Seriously, I thought I was a moron because I just couldn't grasp how to mock a certain interface and why I would want to. Selfishly, I wanted to hate unit testing if just for the fact that *everybody would not shut UP about it*. No silver bullet.

So, apologies for the rambling post. In summary:

- I did not unit test for a long time. Now I do. Unit tests are a great tool, especially during maintenance.
- For existing projects, you can at least go in and write a test that documents a current input and its output. When you change that big mess of murky code in the future, you'll be able to see if you've altered that little snapshot. This is also an excellent way to change the development culture in your company.

Let the flames begin! =)

Share Improve this answer

Follow

answered Nov 15, 2008 at 4:17

Nicholas Piasecki

25.6k ● 5 ● 82 ● 92

very useful, but I think I disagree with you when you say that the end is near – DanielV Mar 31, 2015 at 10:10

When you use MongoDB database in the your application, I feel Unit Testing would even be more useful especially in testing the CRUD operations of your Entities because MongoDB leaves it to the application developer to ensure Foreign Key Referential integrity is upheld. Unlike RDMS databases, MongoDB and a lot of other NoSQL databases leave enforcement of valid associations of data to the application developer. – CS Lewis Nov 27, 2015 at 5:45

12

Unit testing can't be an afterthought, it is *and has to be* something which factors in to your design. I would go so far as to say that even if you never write or call a single unit test, the benefits it has in leading tight component driven software is worth the effort 95% of the time.

Share Improve this answer

Follow

answered Nov 13, 2008 at 9:37

annakata

75.7k ● 18 ● 115 ● 180

7

Writing and running unit-tests is part of a healthy coding process, it is not an addition the customer should have the choice to pay or not pay for.

Testing strategy is a coding issue just as any other: what datastructures to use, variable naming conventions, comment standards, etc.

Share Improve this answer

Follow

answered Nov 13, 2008 at 10:49

JesperE

64.3k ● 22 ● 142 ● 199

1 Your choice of the word "healthy" conjured up an image of an emergency room: "Would you like antiseptic, for an additional fee?" +1 – Adam Liss Nov 13, 2008 at 12:47

Hopefully the coding environment is more like medical research and less like the trauma unit. – JeffO Feb 17, 2009 at 16:49

7

I use unit testing, and tdd, whenever I can. However in my experience for every unit test advocate there are three or more developers who don't really believe writing unit tests is worth the effort, and that it slows down development. However these people tend to keep quiet, so you are unlikely to see many here.

Share Improve this answer

Follow

answered Nov 13, 2008 at 13:22

David Sykes

49.7k ● 18 ● 74 ● 81

7

I like Bob Martin's analogy: imagine you're a surgeon. What would you say to a patient who wanted to pay the surgery but told you to skip washing up ahead of time?

When a client hires me to code they are hiring me as a professional, as someone with the skills and discipline of a professional. My job is to give them "code just works as it should", and **I know** the best way for me to do that is to use TDD.

Share Improve this answer

Follow

answered Nov 13, 2008 at 15:51

Jeffrey Fredrick

4,503 ● 1 ● 27 ● 21

7

After coming onto a couple of projects that were in production but needed major new functionality, one of my bottom lines as a technical lead starting up a project is that unit-tests are a must.

It just costs too much to try and rewrite code that has been written without unit tests. The code is invariably poorly structured (A multi-thousand line web-service all in a single code behind anyone?) and making changes to it (even when it is well structured) without introducing bugs is a really painful process.

This becomes particularly true when a project enters fire-fighting mode (and not having unit tests contributes to projects getting into that state too) - customers are getting grumpy, they've lost faith in the project, few things worse

than being the poor guy trying to get the next fix in without introducing a whole pile of regression bugs, and not even having unit tests.

Those situations can be so easily avoided or at least mitigated by explaining the value of tests up front. Of course there are situations where unit tests aren't so important but they are truly the exception.

So yes - I insist on unit tests and have spent a lot of time fixing the messes made by other developers who relied on their coding skills.

Share Improve this answer

Follow

edited Nov 13, 2008 at 21:39

answered Nov 13, 2008 at 9:29

David Hall

33.1k ● 10 ● 92 ● 129

4

I often use unit testing for complex mathematical algorithms, for example for functions like LineIntersectsLine where you can define some important examples to test. After that, it is easier to control this function. You can simply rewrite/optimize it and test if it still works, or you can add new tests if you encounter bugs and then try to correct these.

Share Improve this answer

Follow

answered Nov 13, 2008 at 9:13

schnaader

49.7k ● 10 ● 108 ● 139

4

I find newer developers benefit more from unit testing than older ones who had to learn from the school of hard knocks of the pitfalls that could make something fail. Unit testing does not lead to good design - it just leads to a design that is more testable. You need to test your code but the way unit testing is preached is dangerous. "It will force you to design your code better", "How can you test whether it is correct?".

I prefer to have well written structured code that when you read it automatically tells you simply what it is trying to accomplish and how it does it. Functions/classes should be small and concise. They should only have one responsibility. Unit tests don't protect against logical errors.

Unit tests give more false positives than anything else particularly when a project is first written. Good design trumps tests - tests should be the verification stage nothing more. I never bought into the testing comes before everything else concept. In my experience this line of thinking favours testability at the expense of extensible code (may be ok for throwaway projects or one off utilities but ironically unit testing isn't as important for these).

Share Improve this answer

Follow

answered Jan 4, 2013 at 23:27

user1950070

41 ● 1 ● 1

3

I agree with Ken that unit testing is part of software development.

About the cost question, it's true than writing the code plus the unit test is longer than writing just the code. However, when you write the code along with its tests - which is called TDD - Test-Driven Development, you end with "*clean code that works*". When you just write the code, then you have to make it work, which can be long and painful.

Another benefit is that you know where you are, as the code that has been written has already been unit-tested.

To answer your question, yes, I'm using unit testing on my projects when it's possible. I write unit tests for all the new code and I strive to for legacy code.

Share Improve this answer

Follow

answered Nov 13, 2008 at 10:52

philant

35.7k ● 11 ● 73 ● 113

3

I'm a big fan of unit testing, mainly because it's completely addicitive - the "code-test-see horrible red bar-fix-test-see lovely green bar" cycle in JUnit seriously gets the endorphins pumping.

Share Improve this answer Follow

answered Nov 13, 2008 at 14:05

nayfnu

47 ● 2

2

My company makes system components for various companies in the Aerospace Industry. The FAA requires Modified Condition/Decision Coverage for Quality level A Safety Critical flight software (DO-178-B) So for verification we do (again from DO-178-B):

Analysis of all code and traceability from tests and results to all requirements is typically required (depending on software level).

This process typically also involves:

> Requirements based test tools
>
> Code coverage analyser tools
>
> > Other names for tests performed in this process can be:
> >
> > > ***Unit testing***

Integration testing

Black box and acceptance testing

Unit testing reveals code errors all the time.

Share Improve this answer

Follow

edited Nov 14, 2008 at 2:58

answered Nov 14, 2008 at 2:53

Paul

198 ● 1 ● 9 ● 22

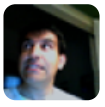
1

In regards to relying on my coding skills, I find that my coding skills actually improve when I use TDD and rely on my unit tests to correct me when I break a test. You learn how to use many language features because you can make a tweak to test out an assumption.

Share Improve this answer

Follow

answered Nov 13, 2008 at 16:20



casademora

69.5k ● 18 ● 71 ● 78

1

I know from experience that when I write code without unit tests, I'll get a few issues that come up to fix, but when I've written the unit tests I rarely hear of issues. Also, from time spent writing unit tests I write better code to begin with. I look at methods and think of the ways they could fail and build them from the start not to fail or at least to fail in a better way.

Unit tests ARE essential to writing better code, but it also will make you a better developer because you SEE areas where things could go wrong and fix them before you ever get to testing.

Share Improve this answer

Follow

answered Nov 18, 2008 at 13:29

xando

202 ● 2 ● 3

0

Unit testing is an essential part of development, and (I have found) will actually reduce the time to completion of a project while improving overall quality, especially when done in a TDD fasion.

Share Improve this answer

Follow

answered Nov 13, 2008 at 10:59

Codebeef

43.9k ● 23 ● 88 ● 119

I do Unit test. I am actully building currently a constraint validation engine and my customers want it bullet proof. Well, without unit test, I would die from stress...

0

So yes, I use it !

Share Improve this answer

Follow

answered Nov 13, 2008 at 12:51

Bite code

595k ● 116 ● 309 ● 334

0

Most functions that I write have tests. Like many have said up there, unit testing is an essential part of software engineering. So, to answer your question, yes, I REALLY write and run tests.

Also, with continuous integration, regression tests will be automated and constantly reported.

Share Improve this answer

Follow

answered Nov 13, 2008 at 13:21

yclian

1,500 ● 15 ● 22

0

Another very useful reminder of why you want to unit test wherever you can just showed up in this post: Top 12 Reasons to Write Unit Tests I need to have that engraved on my retinas.

I can testify personally to the value of thinking about how something will be tested from the beginning, with tests developed through each iteration. I need to be more systematic, myself. I am printing out that list right now.

Share Improve this answer

answered Nov 13, 2008 at 20:11

Follow

orcmid

2,638 ● 19 ● 21

0

Your customers would probably save money overall if unit testing was in place. Some of the errors prevented by unit testing are much more of a liability if found later in the development stage rather than during unit testing. It saves so much headache in the future, now that I use it I don't think I could ever go back.

Share Improve this answer

Follow

answered Nov 13, 2008 at 20:17

Ryan Thames

3,204 ● 6 ● 34 ● 33