The Value of Unit Testing

Asked 15 years, 8 months ago Modified 8 years, 3 months ago

Viewed 2k times

12

Here are some typical answers(ranked in ascending order of corniness) I get from managers/bosses whenever I bring up the importance of having unit tests and code coverage as an integral part of the development cycle

1. "That is the job of QA, just focus on features and development"
2. "The application is not mission critical, if there are some bugs it's not the end of the world"
3. "We can't afford to spend time on unit testing"
4. "Try not to get too fancy"

In spite of having the best intentions of doing a good job, at the end of the day when time comes for the blame game, the burden finally falls on the developer.

It's all too often that I've seen that things break in production, some of which which could have been avoided by catching these bugs statically by running unit tests.

I just wanted to get a conversation going to see what peoples experiences have been and what is the best way

to tackle this.

UPDATE: Thanks everyone for a lot of insightful advice. There are several answers that I wish I could select as the right answer.

c# .net unit-testing

Share

Improve this question

Follow

edited Apr 3, 2009 at 5:09

asked Apr 2, 2009 at 2:17

Am I Crazy

123 ● 5

This is a very good question indeed. One of the best questions here at SO actually. – TobiMcNamobi May 19, 2014 at 10:07

possible duplicate of Is Unit Testing worth the effort? – nawfal Jul 23, 2014 at 15:28

check out my answer, you might wanna change which answer is the best one now :). Actual documented proof, not heir say. – PositiveGuy Sep 17, 2016 at 5:51

12 Answers

Sorted by: Highest score (default)

Introducing unit tests into development process is like investment: you have to put some money up front to get

10

profit later. Management should be more attentive to this analogy if you follow through with it: describe what investments are required and then lay down plan for profits.

For example: Investments:

- time spend to implement test infrastructure (no serious product unit tests can be possible without test-specific infrastructure code that streamlines product specific test patterns, test data creation/removal, etc.);
- time spend on writing actual tests;
- time spend on reviewing and supporting tests;
- etc.

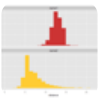
Profits:

- no bug ever re-appears without a sign;
- no major features are released without unit tests passing;
- cycle development-qa-fix bugs is cut in half for majority of bugs: development-unit test-fix bugs;
- etc.

Share Improve this answer

Follow

answered Apr 2, 2009 at 2:44



topchef

19.8k ● 9 ● 67 ● 109

Most managers won't see the advantages of unit testing until they see it in action where it makes sense, so my advice, based on experience, is to take the ff steps:

1. **Apply unit tests to recurring bugs** - This is the best use case to prove the value of unit tests. When you have bugs that just appear and reappear every other build, the unit test will allow developers to see which changes caused the bugs, aside from alerting them in advance that a fix is in order. It's quite easy to demonstrate to management as well.
2. **Apply unit tests to regular bugs** - With the usefulness of unit tests now clearly demonstrated, several instances of recurring bugs disappearing in the long term should be enough to encourage everyone to use unit tests to evaluate *all* bugs, to prevent them from becoming recurring bugs.
3. **Apply unit tests to new functionality** - With unit tests making sure that old bugs don't reccur, and confirm that they are fixed in the first place, the next step would be to apply it to new functionality to ensure that bugs will be minimized. Make it clear that it is impossible to totally eliminate bugs.
4. **Apply full blown TDD** - The final step will be to apply unit testing even before coding, as a design tool that both helps in designing code and minimizing bugs.

Of course I'm not saying that this is easy -- what I had stated above is an oversimplification which even I struggle with everyday -- it's difficult to convince everyone.

If later you decide to move on to a different company, you may want to explicitly look for a company that practices TDD.

Share Improve this answer

Follow

answered Apr 2, 2009 at 3:15

Jon Limjap

95.3k ● 15 ● 103 ● 153

7

People listen to their wallets. Show how much time you can save by catching bugs early on. Translate that to dollar-savings.

Share Improve this answer

Follow

answered Apr 2, 2009 at 2:20

Jamie

6,114 ● 1 ● 19 ● 15

1 +1: Unit test anyway. Don't waste time justifying or explaining. Just do it and do better, less-expensive development. – S.Lott Apr 2, 2009 at 2:32

6

With regard to #3, spending time on unit testing will most likely decrease overall time to market. Great article -

http://blog.scottbellware.com/2008/12/does-test-driven-development-speed-up.html

Share Improve this answer

Follow

answered Apr 2, 2009 at 2:20

Mark Sherretta

10.2k ● 4 ● 38 ● 42

5

For me, the best advantage to adopt the unit test is that I can change my coding behavior to make it more testable, in another word, in more loose coupled way.

if you cannot practice unit test in your real project due to the management issues, I would choose to practice on some small toy project, just to force yourself to get a way to write testable code even there is NO unit test.

My own 2 cents.

Share Improve this answer

Follow

answered Apr 2, 2009 at 2:26

J.W.

18.2k ● 7 ● 45 ● 76

> That's a great point. I practice unit testing on code I write in my own time and it rocks. For me personally it's helped me break dependencies and ensure that my code is designed to be flexible and extensible. Not having that tool at my disposal leaves me handicapped in some ways – Am I Crazy Apr 2, 2009 at 2:32

If unit testing, I'm assuming you're talking about TDD here, is that important to you you should use your own

3

time to write them (assuming you have time). If you do, keep a record of how much time you actually spend writing them and after they've been in place for a release cycle or two go to your managers with some data.

If the answers you've posted are really what your managers are saying then you work for idiots and perhaps some hard data can sway them. Given the market, quitting isn't likely an option and playing office politics won't get you anywhere (or improve the quality of your code).

Until your managers understand that TDD **IS NOT** solely about preventing bugs or "testing" they will NEVER get it. TDD is about design and overall code quality.

You have to show them. If they can't be persuaded then I would start looking. Quietly ;)

Share Improve this answer Follow

answered Apr 2, 2009 at 2:50

user1921

+1 - I'm not sure he's really talking about introducing TDD per se. However, I'm glad that you did. – Mark Brittingham Apr 2, 2009 at 2:54

Mark is right. I wasn't referring to TDD in particular.Just the idea of unit testing your code – Am I Crazy Apr 2, 2009 at 3:09

My short and incomplete advice would be to:

2

- Just change jobs. A company whose managers give that sort of answers is going to fail anyway, and soon. Get out before it's too late.
- Reverse the blame game. Make an official statement every time something gets released without unit testing that it has been done so, and that you are not guaranteeing it's bug-free.
- Write down the time you spend on tasks, *separating bug fixing after failed deployments*, and total it against a (potential) time allotment for writing unit tests.

Share Improve this answer

Follow

answered Apr 2, 2009 at 2:41

Varkhan

16.7k ● 5 ● 32 ● 24

1 +1 on just change jobs, but also make sure that your management and your (former) colleagues know why you're doing it. Don't "enable" idiot managers. – John Saunders Apr 2, 2009 at 4:15

2

Why don't you just write unit tests for your code? Do you know if there are some other developers having the same problem? Probably they will follow your example and write unit tests, too.

I don't think that the problem is the technique or the costs for an integration server. The problem is the

management's attitude to unit testing. So convince them with all developers.

There are lots of hints in this thread (Jon Limjap's answer), try it!

Share Improve this answer

Follow

edited May 23, 2017 at 12:00

Community Bot

1 ● 1

answered Apr 2, 2009 at 5:50

guerda

24k ● 28 ● 100 ● 150

2

Don't sell management on a particular approach; that's just going to be difficult and isn't really going to buy you much. Whether or not your management chain appreciates unit tested code doesn't matter.

Sure, unit testing your code has a lot of benefits associated with it, but don't rely on management buy-off to write your tests. When people start seeing results, they'll flock towards The Right Thing.

Share Improve this answer

Follow

answered Apr 2, 2009 at 6:02

Gabriel Isenberg

26.3k ● 4 ● 38 ● 58

One other thought to add to the other excellent comments on this thread (many of which I've upvoted): make sure that your management knows that Unit testing is very

1

highly automated at this point. I find it very impressive to pop NUnit on the screen, hit the "Run All" button and see dozens of green-lighted tests being passed in seconds. Do that once, saying "this verifies that all of my older work is still correct despite all of my newer changes" and you just may win a few converts. In any event, they'll come to trust you - with your visible proof of quality - more than they trust others. That can only be good for your career.

Share Improve this answer

Follow

answered Apr 2, 2009 at 2:52

Mark Brittingham

28.9k ● 12 ● 82 ● 111

I did try that, in one instance using Unit tests to hunt down perf. issues & fix them,something which would have been hard if not impossible using traditional dev. techniques. To a large degree it's the work culture. Slap something together and move on to the next thing. Fix breakage in real time! – Am I Crazy Apr 2, 2009 at 3:07

Wow, that's really sad. Good luck if you decide to look for better work elsewhere! – Mark Brittingham Apr 2, 2009 at 16:44

1

Well the classical response is that the earlier you catch a bug the less expensive it is to fix, I think most managers can relate to that.

As Mark said showing something concrete is the best way to convince PHBs that something is good as they are so used to hearing talk and probably don't know the difference between unit testing and other testing.

Share Improve this answer

Follow

answered Apr 2, 2009 at 2:55

AndersK

36.1k • 6 • 63 • 86

@Anders - "don't know the difference between unit testing and other testing".That is so true. I got into an argument today when a said PHB was justifying that QA testing is sufficient for testing a framework I'm working on! A true developer would cringe at this answer. Grrrr.... – Am I Crazy Apr 2, 2009 at 3:14

0

Now there is a resource to help. A modern list of use cases, tangible evidence for TDD.

Do you need to convince your boss or teammates that TDD is being used? That it's not some theory? That it's not just heir say?

Now you can, check out WeDoTDD.com, a list of companies that TDD and the stories behind those teams.

That's exactly why I created this site, to put to rest the arguments around "TDD proof" and "Does TDD work" and "Who is doing TDD".

You can also learn a lot about the topic itself there by reading the stories behind these companies and teams practicing it.

Share Improve this answer

Follow

answered Sep 17, 2016 at 5:50

PositiveGuy
20k 30 92 151