How do you persuade others to write unit tests? [closed]

Asked 15 years, 11 months ago Modified 11 years, 4 months ago

Viewed 3k times

14

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 12 years ago.

I've been test-infected for a long time now, but it would seem the majority of developers I work with either have never tried it or dismiss it for one reason or another, with arguments typically being that it adds overhead to development or they don't need to bother.

What bothers me most about this is that when I come along to make changes to their code, I have a hard time getting it under test as I have to apply refactorings to make it testable and sometimes end up having to do *a lot* of work just so that I can test the code I'm about to write.

What I want to know is, what arguments would you use to persuade other developers to start writing unit tests? Most developers I've introduced to it take to it quite well, see the benefits and continue to use it. This always seems to be the *good* developers though, who are already interested in improving the quality of their code and hence can see how unit testing does this.

How do persuade the rest of the motley crew? I'm not looking for a list of testing benefits as *I* already know what these are, but what techniques you have used or would use to get other people on board. Tips on how to persuade management to take an active role are appreciated as well

unit-testing

Share Follow

edited Aug 1, 2013 at 13:34

user1228

asked Jan 6, 2009 at 11:50

tddmonkey

21.2k ● 10 ● 59 ● 69

Very similar to this: stackoverflow.com/questions/67299/… – guerda Jan 6, 2009 at 11:55

test-infected: junit.sourceforge.net/doc/testinfected/testing.htm – philant Jan 6, 2009 at 14:17

13 Answers

Sorted by: Highest score (default)

8

There's more than one side to that question, I guess. I find that actually convincing developers to starting using tests is not that hard, because the list of advantages of using testing often speaks for itself. When that said, it is quite a barrier to actually get going and I find that the learning curve often is a bit steep – especially for novice coders. Throwing testing frameworks, TDD test-first mentality, and mocking framework at someone who's not yet comfortable with neither C#, .Net or programming in general, could be just too much to handle.

I work as a consultant and therefore I often have to address the problem of implementing TDD in an organization. Luckily enough, when companies hire me it is often because of my expertise in certain areas, and therefore I might have a little advantage when it comes to getting people's attention. Or maybe it's just that it's a bit easier to for me as an outsider to come in to a new team and say "Hi! I've tried TDD on other projects and I *know* that it works!" Or maybe it's my persuasiveness/stubbornness? :) Either way, I often don't

find it very hard to convince devs to start writing tests. What I find hard though, is to teach them how to write good unit tests. And as you point out in your question; to stay on the righteous path.

But I have found one method that I think works pretty well when it comes to teaching unit testing. I've blogged about it here, but the essence is to sit down and do some pair-programming. And doing the pair programming I start out writing the unit test first. This way I show them a bit how the testing framework work, how I structure the tests and often some use of mocking. Unit tests should be simple, so all in all the test should be fairly easy to understand even for junior devs. The worst part to explain is often the mocking, but using easy-to-read mocking frameworks like Moq helps a lot. Then when the test is written (and nothing compiles or passes) I hand over the keyboard to my fellow coder so that (s)he can implement the functionality. I simply tell her/him; "Make it go green!" Then we move on to the next test; I write the test, the 'soon-to-be-test-infected-dev' next to me writes the functionality.

Now, it's important to understand that at this point the dev(s) you are teaching are probably not yet convinced that this is the right way to code. The point where most devs seem to see the (green) light is when a test fails due to some code changes that they never thought would break any functionality. When the test that covers that functionality blows up, that's when you've got yourself a

loyal TDD'er on your team. Or that's at least my experience, but as always; your mileage will vary :)

Share Follow

answered Jan 6, 2009 at 14:02

Kjetil Klaussen

6,366 ● 1 ● 37 ● 29

I like the "make it go green" idea. I'm embarking on writing some training courses so may well include this approach in that to get them thinking about what they're doing rather than just copying what I'm doing. Thanks! – tddmonkey Jan 6, 2009 at 14:58

I'm seconding the suggestion of pair programming as a teaching method. Pair programming is how I finally learned to do TDD the right way, and I haven't looked back since. And it doesn't need to be at all formal. Our scrum master asked if he could sit with me one day and did some back seat driving as I coded. "Shouldn't you write that test first? Why are you writing so many tests? You seem to be writing a lot of code without a test around it." – haydenmuhl Jul 6, 2011 at 15:12

7

Quality speaks for itself. If you're more successful than everyone else, that's all you need to say.

Share Follow

answered Jan 6, 2009 at 11:56

S.Lott

391k ● 82 ● 517 ● 788

7

Use a test-coverage tool. Make it very visible. That way everybody can easily see how much code in each area is passed, failed and untested.

Then you may be able to start a culture where "untested" is a sign of bad coding, "failed" is a sign of work in progress and "passed" is a sign of finished code.

This works best if you also do "test-first". Then "untested" becomes "you forgot step 1".

Of course you don't need 100% test coverage. But of one area has 1% coverage and another has 30%, you have a metric for which area is most likely to fail in production.

Share Follow

answered Jan 6, 2009 at 12:13

myplacedk

1,564 ● 2 ● 14 ● 19

4

Lead by example. If you can get evidence that there are less regression on unit tested code that elsewhere.

Getting QA and management buy-in so that your process mandates unit testing.

Be ready to help others to get started with unit testing: provide assistance, supply a framework so that they can start easily, run an introductory presentation.

Share Follow

answered Jan 6, 2009 at 12:38

philant
35.7k • 11 • 73 • 113

3

You just have to get used to the mantra "if it ain't tested, the work ain't done!"

Edit: To add some more meat to my facetious comment above, how can someone know if they're actually finished if they haven't tested their work?

Mind you, you will have a battle convincing others if time isn't allowed in the estimate for the testing of the devleoped code.

A one-to-one split for between effort for coding and that for testing seems to be a good number.

HTH

cheers,

Rob

Share Follow

answered Jan 6, 2009 at 11:53

Rob Wells
37k • 13 • 84 • 147

I'll add that you have to preach by example as well and don't give up. – CheGueVerra Jan 6, 2009 at 13:55

3

Give compliments for one writes more test and produce good results and show the best one to others and ask them to produce the same or better result than this.

Share Follow

answered Jan 6, 2009 at 12:00

Warrior

39.4k ● 44 ● 142 ● 215

3

People (and processes) don't change without one or more pain points. So you need to finjd the significant pain points and demonstrate how unit testing might help deal with them.

If you can't find any significant pain points, then unit testing may not add a lot of value to your current process.

As Steve Lott implies, delivering better results than the other team members will also help. But without the pain points, my experience is that people won't change.

Share Follow

answered Jan 6, 2009 at 12:52

HTTP 410

17.6k ● 14 ● 79 ● 129

2

Two ways: convince the project manager that unit testing improves quality AND saves time overall, then have him make unit tests mandatory.

Or wait for a development crunch just before an important release date, where everyone has to work overtime and weekends to finish the last features and eliminate the last bugs, only to find they've just introduced more bugs. Then point out that with proper unit tests they wouldn't have to work like that.

Another situation where unit tests can be shown as indispensable is when a release was actually delivered and turns out to contain a serious bug due to last-minute changes.

Share Follow

edited Jan 6, 2009 at 12:31

answered Jan 6, 2009 at 12:15

Michael Borgwardt

346k ● 80 ● 486 ● 723

My PM is fully onboard with unit testing as he's seen the benefits elsewhere, but it's persuading his boss that is a stumbling block, with too much lip service being paid but no commitment when it comes down allocating resources.
– tddmonkey Jan 6, 2009 at 12:20

Also, we're required to do unit tests for SOX compliance but I'd rather be persuading developers it's a good idea *because* it's a good idea, not because it's something that's mandated with no good reasons to back it up – tddmonkey Jan 6, 2009 at 12:21

I've been in that same situation and made it work by giving a good example and pointing out situations where unit tests would have saved time and effort - it took quite some time,

but eventually the developers were convinced.
– Michael Borgwardt Jan 6, 2009 at 12:29

1

If developers are seeing that the "successful" developers are writing unit tests, and they are still not doing it then I suggest unit tests should become part of the formal development life-cycle.

E.g. nothing can be checked in until a unit test is written and reviewed.

Share Follow

answered Jan 6, 2009 at 12:00

Joe Ratzer
18.5k ● 3 ● 38 ● 52

1

Probably reefnet_alex' answer helps you: Is Unit Testing worth the effort?

> I think it was Fowler who said: "Imperfect tests, run frequently, are much better than perfect tests that are never written at all". I interprate this as giving me permission to write tests where I think they'll be most useful even if the rest of my code coverage is woefully incomplete.

Share Follow

edited May 23, 2017 at 11:48

Community Bot
1 ● 1

answered Jan 6, 2009 at 11:57

guerda

24k ● 28 ● 100 ● 150

1

You mentioned that your manager is on board with unit tests. If that's the case, then why isn't he (she) enforcing it? It isn't your job to get everybody else to follow along or to teach them and in fact, other developers will often resent you if you try to push it on them. In order to get your fellow developers to write unit tests, the manager has to emphasize it **strongly**. It might end up that part of that emphasis is education on unit test implementation which you might end up being the educator and that's great, but management of it is everything.

If you're in an environment where the group decides the style of implementation, then you have more of a say in how the group dynamic should be. If you are in that sort of environment and the group doesn't want to emphasize unit tests while you do, then maybe you're in the wrong group/company.

Share Follow

answered Jan 6, 2009 at 14:27

shank

484 ● 2 ● 5

I think he's of the same opinion as me that **enforcing** it just builds resentment towards doing it. I think it's doubly hard for me as I'm *just a contractor* and so struggle to get my opinions heard above the long standing permies. – tddmonkey Jan 6, 2009 at 15:53

I didn't realize you are a contractor. You're in a tough spot then. Even as a "permie" a person can only tell their boss the same thing X number of times. Once you've gotten your point across and made the benefits clear, you've done all you can. After that, it starts to sound like complaining. – shank Jan 7, 2009 at 0:54

1

I have found that "evangelizing" or preaching rarely works. As others have said, do it your way for your own code, make it known that you do it, but don't try to force others to do it. If people ask about it be supportive and helpful. Offer to do a few lunch-time seminars or informal dog and pony shows. That will do a lot more than just complaining to your manager or the other developers that you have a hard time writing tests for code they wrote.

Slow and steady - it is not going to change overnight.

Once I realized that at one place where I worked the acceptance for peer reviews improved tremendously. My group just did it and stopped trying to get others to do it. Eventually people started asking s about how we got some of the success we did. Then it was easier.

Share Follow

answered Jan 6, 2009 at 14:42

Tim

20.4k ● 24 ● 122 ● 219

We have a test framework which includes automated running of the test suite whenever anyone commits a

0

change. If someone commits code that fails the tests, the whole team gets emailed with the errors.

This leads to introduced bugs being fixed pretty quickly.

Share Follow

answered Jan 28, 2009 at 2:44

noswonky

196 ● 4