Is Unit Testing worth the effort? [closed]

Asked 16 years, 3 months ago Modified 9 years, 11 months ago

Viewed 315k times

572

votes

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.

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

I am working to integrate unit testing into the development process on the team I work on and there are some sceptics. What are some good ways to convince the sceptical developers on the team of the value of Unit Testing? In my specific case we would be adding Unit Tests as we add functionality or fixed bugs. Unfortunately our code base does not lend itself to easy testing.

unit-testing

Share

edited Apr 10, 2013 at 19:43

community wiki
7 revs, 5 users 67%
George Stocker

25 How can you ask such a thing.. Unit testing is so hip :) ? Seriously... the general idea is the same... do unit testing if you feel that benefits outweigh the cost... if you have reasons to believe otherwise.. find something else that helps. – Gishu Sep 29, 2008 at 6:32

14 (regular asserts with debug\release build + clean interface to classes + dedicated dummy tester) > unit testing – Viktor Sehr Sep 30, 2010 at 19:34

110 Having been on a few teams with the same attitude, my experience is that you're wasting your time. The skeptics are just giving you the run-around, and will sabotage and stonewall you until you get frustrated and move to an organization that shares your values. Which is probably what you should do. If the "lead" of the team is one of the skeptics, think seriously about getting out. I've seen this sort of resistance to unit testing as just the tip of the iceberg of bad development practices. – sea-rob Feb 5, 2013 at 18:21

7 @ViktorSehr: unit tests are not in opposition to clean interfaces, in fact they are the opposite when applying TDD. Either that, or driver seat + steering wheel > garage. – Jonas Byström Feb 19, 2013 at 21:05

5 Unit testing is a massive waste of time that rarely ever discovers bugs but eats away 30% of bug-fixing and development time and budget – nanobar Nov 5, 2013 at 9:24

Comments disabled on deleted / locked posts / reviews |

44 Answers

Sorted by: Highest score (default)

2 votes

From my experience, unit tests and integration tests are a "MUST HAVE" in complex software environments.

In order to convince the developers in your team to write unit tests you may want to consider integrating unit test regression analysis in your development environment (for example, in your daily build process).

Once developers know that if a unit test fails they don't have to spend so much time on debugging it to find the problem, they would be more encouraged to write them.

Here's a tool which provides such functionality:

unit test regression analysis tool

Share

edited Dec 7, 2012 at 2:35

community wiki
2 revs, 2 users 96%
Liorp

2
votes

The one thing to keep in mind about unit testing is that it's a comfort for the developer.

In contrast, functional tests are for the users: whenever you add a functional test, you are testing something that the user will see. When you add a unit test, you are just making your life easier as a developer. It's a little bit of a luxury in that respect.

Keep this dichotomy in mind when you have to make a choice between writing a unit or a functional test.

Share

edited Jan 9, 2014 at 1:02

community wiki
2 revs, 2 users 80%
Cedric Beust

1
vote

The whole point of unit testing is to make testing easy. It's automated. "make test" and you're done. If one of the problems you face is difficult to test code, that's the best reason of all to use unit testing.

Share

answered Sep 15, 2008 at 21:47

community wiki
Deathbob

1

vote

When you manually test software, you normally have a small set of tests/actions that you use. Eventually you'll automatically morph your input data or actions so that you navigate yourself around known issues. Unit tests should be there to remind you that things do not work correctly.

I recommend writing tests before code, adding new tests/data to evolve the functionality of the main code!

Share

answered Sep 15, 2008 at 21:53

community wiki
Ray Hayes

1

vote

As a physics student i am very driven to actually *prove* that my code works as it is supposed to. You could either prove this logically, which increases in difficulty drastically as implementation gets more complex, or you can make an (as close as possible) empirical proof of function through good testing.

If you don't provide logical proof of function, you have to test. The only alternative is to say "I think the code works...."

Share

answered Sep 15, 2008 at 21:54

community wiki
Fredrik

1

vote

One of the benefits of unit testing is predictability.

Before unit testing I could have predicted to a great degree of accuracy how long it would take to code something, but not how how much time I would need to debug it.

These days, since I can plan what tests I am going to write, I know how long coding is going to take, and at the end of coding, the system is already debugged! This brings predictability to the development process, which remove a lot of the pressure but still retains all the joy!!.

Share

answered Sep 15, 2008 at 22:09

community wiki
Raz

1

vote

Just today, I had to change a class for which a unit test has been written previously.
The test itself was well written and included test scenarios that I hadn't even thought about.
Luckily all of the tests passed, and my change was quickly verified and put into the test environment with confidence.

Share

answered Sep 30, 2010 at 18:21

community wiki
crowne

1

vote

@George Stocker "Unfortunately our code base does not lend itself to easy testing.". Everyone agrees that there are benefits to unit testing but it sounds like for this code base the costs are high. If the costs are greater than the benefits then why should they be enthusiastic about it? Listen to your coworkers; maybe for them the perceived pain of unit tests is greater than the perceived value of unit tests.

Specifically, try to gain value as soon as possible, and not some feel-goodery "xUnit is green" value, but clean code that users and maintainers value. Maybe you have to mandate unit tests for one iteration and then discuss if it is worth the effort or not.

Share

answered Dec 20, 2010 at 7:50

community wiki
Trystan Spangler

1

vote

Make the first things you test not related to unit testing. I work mostly in Perl, so these are Perl-specific examples, but you can adapt.

- Does every module load and compile correctly? In Perl, this is a matter of creating a Foo.t for each Foo.pm in the code base that does:

```
use_ok( 'Foo' );
```

- Is all the POD (Plain Ol' Documentation) formatted properly? Use Test::Pod to validate the validity of the formatting of all the POD in all the files.

You may not think these are big things, and they're not, but I can guarantee you will catch some slop. When these tests run once an hour, and it catches someone's premature commit, you'll have people say "Hey, that's pretty cool."

Share

edited Feb 21, 2014 at 15:22

community wiki
2 revs, 2 users 95%
Andy Lester

0
votes

Who are you trying to convince? Engineers or manager? If you are trying to convince your engineer co-workers I think your best bet is to appeal to their desire to make a high quality piece of software. There are numerous studies that show it finds bugs, and if they care about doing a good job, that should be enough for them.

If you are trying to convince management, you will most likely have to do some kind of cost/benefit reasoning saying that the cost of the defects that will be undetected is greater than the cost of writing the tests. Be sure to include

intagable costs too, such as loss of customer confidence, etc.

Share

answered Sep 15, 2008 at 21:50

community wiki
Craig H

0

votes

Unit testing help you to release software with fewer bugs while reducing overall development costs. You can click the link to read more about the benefits of unit testing

Share

answered Jun 18, 2009 at 10:41

community wiki
Steve_0

0

votes

This is very .Net-centric, but has anybody tried **Pex**?

I was extremely sceptical until I tried it - and wow, what a performance. Before I thought "I'm not going to be convinced by this concept until I understand what it's actually *doing* to benefit me". It took a single run for me to change my mind and say "I don't *care* how you know there's the risk of a fatal exception there, but there *is* and now I know I have to deal with it"

Perhaps the only downside to this behaviour is that it will flag up *everything* and give you a six month backlog. But, if you had code debt, you always had code debt, you just didn't know it. Telling a PM there are a potential two hundred thousand points of failure when before you were aware of a few dozen is a nasty prospect, which means it is vital that the concept is *explained first.*

Share

answered Sep 30, 2010 at 18:27

community wiki
Tom W

0
votes

Unit Testing is one of the most adopted methodologies for high quality code. Its contribution to a more stable, independent and documented code is well proven . Unit test code is considered and handled as an a integral part of your repository, and as such requires development and maintenance. However, developers often encounter a situation where the resources invested in unit tests where not as fruitful as one would expect. In an ideal world every method we code will have a series of tests covering it’s code and validating it’s correctness. However, usually due to time limitations we either skip some tests or write poor quality ones. In such reality, while keeping in mind the amount of resources invested in unit testing development and maintenance, one must ask himself, given the available time, which code deserve testing the most? And

from the existing tests, which tests are actually worth keeping and maintaining? See here

Share

answered Jan 16, 2012 at 9:19

community wiki
RoiG

-9

votes

Unit testing works for QA guys or your managers, not for you; so it's definitely not worth it.

You should focus on writing correct code (whatever it means), not test cases. Let other guys worry about those.

Share

answered Feb 10, 2009 at 0:59

community wiki
Mark

5 -1: Not true. As a developer, you should always worry about breaking existing functionality. When done right, automated unit tests can make you feel more confident that your new code doesn't break existing features. – Jim G. Sep 30, 2010 at 19:20

4 Meaningless statement. How you intend to determine whether or not your code is 'correct' without testing it? – Tom W Sep 30, 2010 at 20:28

2 Go get'em Tom W! Unit testing is the answer to your "whatever it means". This comment is a big, huge, gigantic fail.
– Ryan Cromwell Oct 4, 2010 at 14:56

one of the worst SO answer ever. – Teoman shipahi Apr 30, 2014 at 21:41

Prev 1 2