How to prove writing specifications beats code cowboys? [closed]

Asked 15 years, 10 months ago Modified 14 years, 9 months ago

Viewed 1k times

12

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 7 years ago.

Improve this question

So I have a problem. Or rather my friend has a problem, since I would never write about my company on an internet forum.

At my friend's company specification writing is, shall we say, a little underused. There's a deeply ingrained culture of writing code first and asking questions later, whether it's for a library routine or a new tool to inflict on their long suffering designers.

This of course leads to situations where functionality is partially correct, incorrect, or just completely missing ("oh, just save before trying anything you may want to undo").

This usually results in a loss of productivity for those poor designers, or beta periods where bug-fixing is largely spent implementing things correctly.

My friend's found his suggestions of writing (and testing against) specifications to be generally well received. Most of his colleagues have embraced the wonderful feeling of discovering false-assumptions on paper, instead of at 11pm on a Sunday in the middle of beta. Viva La Revolution!

However there are a few who poo-poo anything that stands between their task and a keyboard. They laugh at the thought of actually designing anything, and write code with merry abandon. Mostly these are senior, long employed developers, reluctant to "waste time".

The problem is that this second group of heretics invariably produce things (or at least something) quicker than the first. Subsequently this becomes justification along the lines of "It's pointless to write specifications for something as simple as an image resizer! Oh and those bugs where width!=height or the image uses RLE just need a few tweaks".

And now the question :)

Other than saying "told you so" at the end of a project, what are some good short-term ways to demonstrate how the practice of writing functional or technical specifications leads to better software in the long run?

Cheers!

specifications

Share

Improve this question

Follow

edited Mar 5, 2009 at 19:47

asked Feb 23, 2009 at 22:27

Andrew Grant

58.8k ● 22 ● 131 ● 144

Why is this flagged for closed? – Paul Nathan Feb 23, 2009 at 22:42

This isn't a forum, But you already knew that, Andrew. :-) – George Stocker Feb 24, 2009 at 0:04

if you want to edit to clarify something or correct typos/spelling then please do so. However do not rewrite portions of my post. It's rude and uncalled for. – Andrew Grant Feb 24, 2009 at 1:50

Wow, you deleted all of my comments detailing why the edits were made. – George Stocker Mar 5, 2009 at 20:01

11 Answers

Sorted by: Highest score (default)

15

the goal is to write the right piece of software.... specifications is just one tool to try and find / define what the right thing is.

I think as a group it needs to be decided how you go about building the right thing and how you communicate that amongst everyone.

ie, focus on the real problem and then get shared agreement about how to solve it. Rather than say, "here's a solution to a problem, lets do it!", that often doesn't get a lot of buy in from everyone. It could also be that specifications are not quite the right thing to solve the problem either.

Share Improve this answer

Follow

answered Feb 23, 2009 at 22:36

Keith Nicholas

44.3k ● 15 ● 99 ● 166

> Good point, it's always a good idea to use the right tool for the job. Some people's toolboxes only have hammers though : (– Dana the Sane Feb 23, 2009 at 22:45

10

The one area where specifications beat cowboys is "scope creep" and the costs associated with it.

The specifications is a contract between the customer and developers. If you do not have a clear contract, how would you know when the contract has been satisfied?

Also, having detailed specifications makes it easier to have two or more developers working on different parts of the project at the same time. And, it gives the testers something to compare the software to when making sure the damn thing works!

Share Improve this answer

Follow

edited Feb 23, 2009 at 22:59

answered Feb 23, 2009 at 22:50

BoltBait

11.5k ● 9 ● 60 ● 91

5

This is pretty difficult since office culture and work habits are very hard to change. If you are really serious about this though, try to get management to agree to a trial where specs are used for a small project/module, and maintenance costs (time, bugs, etc) are quantified over time.

You may not get the other developers on your side this way, but $$'s are easier to understand than more abstract dev practices. The management of the poo-poo'ers are responsible for that group's continued employment and performance reviews, so that is one way to convince them to change, or to have them work somewhere else.

Share Improve this answer

Follow

answered Feb 23, 2009 at 22:32

Dana the Sane

15.2k ● 8 ● 60 ● 81

3

If you have the power, try having all developers, when they have any questions about someone else's routines, to ask the original coder directly. The few coders who still believe that specs and docs aren't important might be

willing to try it a few times if they're stuck answering "newbie questions" all the time.

This was exactly the scenario I worked under at a large dev house some years back, and people learned quickly that they either had to heavily document their code, or risk "wasting their time" responding to a lot of small issues.

Of course, this is only feasible if you can get enough people to try it :)

Share Improve this answer

Follow

answered Feb 23, 2009 at 22:40

Mike

4,590 ● 1 ● 28 ● 29

1

I think it might help if you would write the specs with the whole team. Take the time to do it together and discuss things in group. We usually have such meetings at the beginning of every iteration with the whole team and if needed, we discuss more details with a few developers. It is not necessary in my experience to discuss every single detail because that's where it gets boring for some people.

Share Improve this answer

Follow

answered Feb 23, 2009 at 22:37

Christophe Herreman

16.1k ● 9 ● 60 ● 86

It usually takes time - and that is always the problem.

1

EDIT:

For your own company, I mean your friend, needs to document whenever something had to change and a spec was not available. Whenever a spec or docs would have saved time, record that event. Record the amount of time spent (both by the person researching and the person (if any) who was asked, helped, etc. Then you need to do a cost/benefit analysis of whether the time "saved" initially by not doing a spec was worth not having it when it was needed.

In some cases it probably pays off to write one, and you may find other cases where it is not "needed". In practice, the payoffs are potentially so big that it is always best just to have that done in the beginning.

Now - the caveat:

- You may have reluctant people. Don't try to convince them just yet.
- You need a repository that is standard use - it does not matter where or how, but it HAS to be consistent and well-known so they can be found
- they have to be kept up to date and maintained

edited Feb 23, 2009 at 23:10

answered Feb 23, 2009 at 22:38

Tim
20.4k • 24 • 122 • 219

1

Split the two cultures into two different groups, come up with your preferred performance metrics and set them loose. Best group wins and all that.

Or point them to any one of a thousand different quantitative/qualitative studies that scientifically prove his point.

Or alternatively sack all the cowboys and/or other wise mandate the use of a minimum level of specification regardless of who it annoys. Depending on project complexity you may only realise how detrimental a lack of any specification and documentation really is until most of the original developers have left any way.

Or sub contract an application module to a purely outsourced team and see how detrimental a lack of specs and docs is to their performance and ability to meet key metrics.

Or ask the customers about product quality/cost/etc.

If their employment contracts were properly designed and enforced your friend wouldn't be stuck telling developers "i told you so" at the end of the project but could instead give them their pink slips.

Besides all that, you shouldn't have to wait till end of a project to know there is a problem nor to make personel

changes.

Or advise your friend to quit and join a shop that has development methods and practices more in line with his preferences. "Maybe he's just in the wrong place."

Share Improve this answer

Follow

answered Feb 24, 2009 at 1:37

rism

12.1k ● 16 ● 78 ● 121

1

A specification is about knowing what done is. How can you know its time to go home if you don't know what done is?

In the simplest case, do developers talk with stakeholders to get a feel for what they want? If so, then whatever they decide on is the 'spec'.

White that down -- implement it -- and go home.

Share Improve this answer

Follow

answered Feb 24, 2009 at 1:48

Nathan Feger

19.5k ● 11 ● 64 ● 73

0

Just, tell your friend they have to avoid going to the other extreme where nothing could be done until the spec is perfect and 100% complete . And the coding delayed each week because someone added something else to the spec. There will be times where that approach could take 2 - 3 months and after which someone at the high

management level will get very upset and will say "I don't care, I just want the product done" and the situation will be worst.

You can get the best of both worlds by keeping the flow agile (weekly review, deliver early, quick look ups etc.)

Share Improve this answer

Follow

answered Feb 23, 2009 at 23:19

OscarRyz

199k ● 119 ● 396 ● 573

0

Specs IS the waste of time. Most of the people who will use the software can't tell what they want but they can tell if what they get is good enough or not.

Prototyping rarely works for the same reason, they can't tell if this feature is not done yet or it won't be done at all, so they kind of go ballistic telling you it's all wrong and you should redo everything when actually they need just a quick tweak. Or they would tell you it's all good up to the release point when they finally would realize it's not usable.

The best way to design is to go out there and see how they used to do things and try to fit your application into theirs work flow. EVERY team member should do that.

I'm not saying you can't write requirements down but showing them to your actual users has no point. And you can very well hold all the spec in your head if you working on the feature alone.

In brief: don't write specs, spy on your users. Make sure all team members do that.

Share Improve this answer

Follow

answered Feb 24, 2009 at 1:30

vava

25.4k ● 11 ● 66 ● 79

0

Spending too much time on specs *is* waste of time. I once worked on a project where a one-off Professional Services app was sold, written and shipped, after which the bosses demanded that we write specs. The thing is, it took longer to write the spec, required more people, and cost more than the original app, which will never, ever be used again.

Agile methodology says that we shouldn't burden ourselves by going crazy with documentation. Instead, *requirements* should be clearly delineated and assigned to individual teams and developers, with regular code reviews. This works great when you have a competent architect on board, but can be disastrous if project leadership are not qualified to determine the relative technical merit of different approaches advanced by different parts of the team.

Code should be self-documenting. Code standards should be adhered to. Requirements should be carefully thought out and enumerated. But what you do not need is a 200 page document describing what happens on every freakin' line of code in every module, or comments that are more detailed than the code itself. Developers should

be able to read this stuff. If they can't, your problem lies elsewhere.

Share Improve this answer

Follow

answered Mar 16, 2010 at 21:00

3Dave

29k ● 18 ● 90 ● 151