How do you argue for the "high notes", the true production class code?

Asked 15 years, 10 months ago Modified 13 years, 5 months ago

Viewed 380 times

2

This question has been addressed in a similar post on a more specific level regarding UI.

I would like to address the subject on a more general design level.

I make descisions on design every day to ensure high quality. But every now and then I get into dicsussions with middle management and unexperienced developers about the gain of doing things "the right way".

Sometimes I just say "trust me, I've seen where this leads to, we're doing it the other way", sometimes I make an attempt lay down a scenario where a particular set of choices introduce problems etc. Most of the time I feel I'm not reaching the person I'm talking to. I might as well have said "trust me".

I feel that one of my capabilities as a senior software guy should be to explain and motivate the technical choices we make as a company. I can do that on economical and user experience level.

But I seem to fail to explain on technical and pseudo-technical levels why some design choices "feels wrong" and why others just feels more correct and benefitial, even though at first it might be harder to implement or seem unnecessarily complex.

Luckily, I occasionally show good results, otherwise I probably would begin to doubt the whole concept of good vs. bad design.

I would really find it interesting to read about what others here have to say about this.

Thanks in advance!

software-quality

Share

Improve this question

Follow

edited May 23, 2017 at 11:47

Community Bot

1 ● 1

asked Feb 10, 2009 at 14:22

sharkin

12.5k ● 24 ● 89 ● 122

1 What on earth is "true production class"? – Jay Bazuzi Feb 10, 2009 at 15:40

A rather bombastic term for code that works in the long run I guess :-) – sharkin Feb 10, 2009 at 16:04

9 Answers

Sorted by: Highest score (default)

10

Have you tried the Technical Debt argument? Citing the link (from Martin Fowler site):

> ... doing things the quick and dirty way sets us up with a technical debt, which is similar to a financial debt. Like a financial debt, the technical debt incurs interest payments, which come in the form of the extra effort that we have to do in future development because of the quick and dirty design choice...

Share Improve this answer Follow

edited Feb 10, 2009 at 14:53

answered Feb 10, 2009 at 14:42

mouviciel
67.8k ● 13 ● 108 ● 142

Thanks! This is a brilliant example of my pain - how do you argue that a debt is produced from "quick and dirty ways". The path leading to the debt, what is it? – sharkin Feb 10, 2009 at 14:46

Another citation from the article is: "The tricky thing about technical debt, of course, is that unlike money it's impossible to measure effectively." I found reading the whole article, and others linked articles, very instructive. – mouviciel Feb 10, 2009 at 14:50

Measuring the debt is one thing, but if you cannot describe/motivate the chain of events that produce the debt, how can you prove its existance at all? – sharkin Feb 10, 2009 at 14:58

My thought is that only experience can give an insight of it: when I waste several days working around dirty code, I feel how hard it is paying technical debt. And later I rembember this situation and try to avoid writing dirty code myself. – mouviciel Feb 10, 2009 at 15:08

8

Here is my cynical answer:

There are two kinds of people - those who care about doing things the right way and those who do not. You don't need to win over those who agree with you so the folks you have to worry about are those who don't care how things get done. You will most likely find that the folks that don't care about doing things "the right way" care deeply about other things like up-front time penalties, confusion with a different implementation, etc. The best thing you can do is to determine what it is that the other folks really care about and address them.

For instance if folks are worried that you will introduce too much development time up front by doing things the right way then show them how doing things the right way might save them time in the long run.

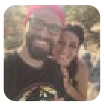
One word of caution though - always ask yourself the ever-important question that all purists (myself included) must always ask whenever they incounter opposition to "the right way":

> **Am I doing this "the right way" for its own sake?**

If the answer is "yes" then you probably need to back down.

Share Improve this answer Follow

answered Feb 10, 2009 at 14:42



Andrew Hare
351k ● 75 ● 645 ● 641

2

Two points:

1. If you are failing to express your rationale to others, it is likely that *yourself* don't understand it enough. So an attempt to make it communicable to others may often-times shine light on your own assumptions which could be wrong. Therefore, no matter how difficult (and I know it is), I'd suggest you make an active effort to crystallize your ideas for others. Write things down, make diagrams -- whatever. Even if you still don't succeed in communicating, you may indeed succeed in seeing error in your judgment.
2. It may be that people you're talking to are not familiar with software engineering very much. In that case, instead of pushing across high-level ideas, you may want to walk them through the first principles just for

once. Create a presentation of sorts (again with diagrams and all) explaining key ideas upon which your thinking hinges and hold a session. You may need to work a bit hard for this initial session. However, if they get it, for the rest of your life, you may have a lot easier time.

Share Improve this answer Follow

edited Feb 10, 2009 at 14:50

answered Feb 10, 2009 at 14:39

G S
36.7k ● 23 ● 89 ● 121

Thanks for the input. I feel I understand it enough for myself, and as I pointed out I can measure the quality of my descisions through results. But I find it hard to explain complex chains of events relating to design, which eventually leads to a problem. – sharkin Feb 10, 2009 at 14:44

1

Often, discussions like these can devolve into religious or political arguments - everyone has an opinion, but no one has the answers. Anything you can do to **add substance to your thoughts** will help you push your ideas through.

Read some existing material - About Face, The Design of Everyday Things, etc. - and see if you can't identify some facts to back up your feelings that some things just feel wrong.

Share Improve this answer Follow

answered Feb 10, 2009 at 14:41

David Koelle
20.8k ● 23 ● 94 ● 130

1

Generally, I've found that if I have to use the argument of "Trust Me. I know what I am talking about" I need to go back and rethink what I am trying to explain and implement. Sometimes I get a feeling about something being good or bad, but that is an indicator that I need to review it as I don't have a concrete reason about why I should or shouldn't do something.

The other thing you should pay attention to is dismissing "Junior" people because of their title. You may have more expierence, but that doesn't mean they aren't smart and could be correct in what they are saying. In reality, years of expierence can mean little to nothing as it is just the number of years that someone has been doing the same type of job. It's a good indicator as to where someone is in their career, but it isn't an automatic decree they are actually capable of making the right desicions all the time.

Share Improve this answer Follow

answered Feb 10, 2009 at 14:46

kemiller2002
115k ● 28 ● 199 ● 253

1

Maybe you shouldn't argue **technical** decisions with middle management: that would be discussing at the wrong level of abstraction. For example, instead of answering "it will take X days of refactoring and then Y days to implement the new feature", just say "it'll take (X+Y) days".

As for arguing with junior developers, you seem to be asking about rhetoric or "how to argue"; that's a huge topic, for example one of many techniques is "appeal to authority" e.g. say "I read about this technique in < *this good book by respected author* >" ... or, even if you can't prove something, argue on the basis that it reduces risk ...

Most importantly though, **listen**: find out why they want to argue something else, and address **their** argument. For example, maybe they want to do it the 'wrong' way because they want it quickly, because they're under some time pressure from someone else: if so, address **that** concern.

Share Improve this answer Follow

answered Feb 10, 2009 at 14:56

ChrisW

56k ● 14 ● 120 ● 233

1

I try to think of it in terms of cost. In a way, it comes back to ["fast, good, cheap: pick any two"](). If you consistently choose fast and cheap, you will have crap.

I prefer to design and to build for three or four years out for any given chunk of functionality. If I have that latitude and can be thorough in my explanation of the need and benefits of the extra effort, everything will get better.

Let me give a concrete example. I was assigned a feature to add to a chunk of code that takes a collection of images and generates a PDF from the images. The existing code was brief and terse and had been written to fill a customer need - but it wasn't built to grow. I could, and did, add the feature using the existing code and checked it in, but it was ugly and every subsequent feature would get uglier. Then I spent an additional two weeks writing a full functional model of generative PDF, architected to also be a consumer as well. After I was done, it took a couple hours to reimplement the entire PDF generator in the new model. I could explain very concisely that building the model, though not strictly necessary, created an enabling technology and would make the cost of adding more features to the existing code cheaper. We now have four major products or feature sets that are entirely based on this technology and are looking at more. Slam dunk.

Share Improve this answer Follow

edited May 23, 2017 at 12:26

Community Bot
1 ● 1

answered Feb 10, 2009 at 15:23

plinth
49.1k ● 11 ● 83 ● 123

1

My boss isn't a very technical person but he also has trouble accepting "trust me" answers. It took me a while to figure out that in order to relate a problem to him, it was important to quantify the issue in terms of time costs, man hours, money spent, etc.

Relating to what the person understands will usually get them to accept your answers a lot more readily.

Share Improve this answer Follow

answered Feb 10, 2009 at 15:36

Soviut
91.4k ● 53 ● 205 ● 279

0

I am not a good salesman either, which seems to be your problem. What I have found that works very well for me when I want to convince people to take another route is to ask *leading* questions. Try to ask questions in such a way that the only logical answer is the one you are looking for. This has the added bonus that the person believes they are coming up with the solution themselves which means they are more likely to believe in the solution. It takes practice, but is easier to learn than to become a salesman when it is not your personality type. It also means that you really need to know your stuff in order to know how to ask your questions that lead to the *correct* solution.

This approach also has a side benefit in that if they can come to different solutions even in the face of your leading questions, then this should cause you to re-examine your own solution.

edited Feb 10, 2009 at 15:08

Share Improve this answer

Follow

answered Feb 10, 2009 at 15:02

Dunk

1,734 ● 1 ● 14 ● 19