Which is better: shipping a buggy feature or not shipping the feature at all?

Asked 16 years, 3 months ago Modified 15 years, 6 months ago

Viewed 928 times

12

this is a bit of a philosophical question. I am adding a small feature to my software which I assume will be used by most users but only maybe 10% of the times they use the software. In other words, the software has been fine without it for 3 months, but 4 or 5 users have asked for it, and I agree that it should be there.

The problem is that, due to limitations of the platform I'm working with (and possibly limitations of my brain), "the best I can do" still has some non-critical but noticeable bugs - let's say the feature as coded is usable but "a bit wonky" in some cases.

What to do? Is a feature that's 90% there really "better than nothing"? I know I'll get some bug reports which I won't be able to fix: what do I tell customers about those? Should I live with unanswered feature requests or unanswered bug reports?

release release-management

Share

Improve this question

Follow

edited May 30, 2009 at 9:58

Ludwig Weinzierl

16.6k ● 10 ● 46 ● 49

asked Sep 17, 2008 at 22:28

Peldi Guilizzoni

389 ● 4 ● 11

15 Answers

Sorted by: Highest score (default)

12

Make sure people know, that you know, that there are problems. That there are bugs. And give them an easy way to proide feedback.

What about having a "closed beta" with the "4 or 5 users" who suggested the feature in the first place?

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:34

Rob Wells

37k ● 13 ● 84 ● 147

3

There will always be unanswered feature requests and bug reports. Ship it, but include a readme with "known issues" and workarounds when possible.

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:32

L. Mills

486 ● 4 ● 12

2

You need to think of this from your user's perspective - which will cause less frustration? Buggy code is usually more frustrating than missing features.

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:34

Mark Ransom

308k ● 44 ● 416 ● 647

2

Perfectionists may answer "don't do it".

Business people may answer "do it".

I guess where the balance is is up to you. I would be swaying towards putting the feature in there if the bugs are non-critical. Most users don't see your software the same way you do. You're a craftsman/artist, which means your more critical than regular people.

Is there any way that you can get a beta version to the 4-5 people who requested the feature? Then, once you get their feedback, it may be clear which decision to make.

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:36

Dan Harper

1,130 ● 1 ● 10 ● 22

1

Precisely document the wonkiness and ship it.
Make sure a user is likely to *see* and *understand* your documentation of the wonkiness.

You could even discuss the decision with users who have requested the feature: do some market research.

Just because you can't fix it now, doesn't mean you won't be able to in the future. Things change.

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:33

JoshM

41 ● 1 ● 3

1

Label what you have now as a 'beta version' and send it out to those people who have asked for it. Get their feedback on how well it works, fix whatever they complain about, and you should then be ready to roll it out to larger groups of users.

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:33

warp
1,578 ● 15 ● 21

1

Ship early, ship often, constant refactoring.

What I mean is, don't let it stop you from shipping, but don't give up on fixing the problems either.

An inability to resolve wonkiness is a sign of problems in your code base. Spend more time refactoring than adding features.

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:36

Bill K
62.8k ● 18 ● 112 ● 158

1

I guess it depends on your standards. For me, buggy code is not production ready and so shouldn't be shipped. Could you have a beta version with a known issues list so users know what to expect under certain conditions? They get the benefit of using the new features but also know that it's not perfect (use that their own risk). This may keep those 4 or 5 customers that requested the feature happy for a while which gives you more time to fix the bugs (if possible) and release to production later for the masses.

Just some thoughts depending on your situation.

Share Improve this answer

answered Sep 17, 2008 at 22:36

Follow

andypike
551 • 2 • 9 • 16

1

Depends. On the bugs, their severity and how much effort you think it will take to fix them. On the deadline and how much you think you can stretch it. On the rest of the code and how much the client can do with it.

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:39

Mostlyharmless
2,283 • 19 • 28

1

I would not expect coders to deliver known problems into test let alone to release to a customer.

Mind you, I believe in zero tolerance of bugs. Interestingly I find that it is usually developers/ testers who are keenest to remove all bugs - it is often the project manager and/ or customer who are willing to accept bugs.

If you must release the code, then document every feature/ bug that you are aware of, and commit to fixing each one.

Why don't you post more information about the limitations of the platform you are working on, and perhaps some of the clever folk here can help get your bug list down.

Share Improve this answer

answered Sep 17, 2008 at 22:43

Follow

David Turner
5,074 ● 6 ● 28 ● 27

1

If the demand is for a feature NOW, rather than a feature that works. You may have to ship.

In this situation though:

- Make sure you document the bug(s) and consequences (both to the user and other developers).
- Be sure to add the bug(s) to your bug tracking database.
- If you write unit tests (I hope so), make sure that tests are written which highlight the bugs, before you ship. This will mean that when you come to fix the bugs in the future, you know where and what they are, without having to remember.
- Schedule the work to fix the bugs ASAP. You do fix bugs before writing new code, don't you?

Share Improve this answer

Follow

answered Sep 30, 2008 at 8:06

Matt Lacey
65.6k ● 12 ● 93 ● 144

1

If bugs can cause death or can lose users' files then don't ship it.

If bugs can cause the application to crash itself then ship it with a warning (a readme or whatever). If crashes might cause the application to corrupt the users' files that they were in the middle of editing with this exact application, then display a warning each time they start up the application, and remind them to backup their files first.

If bugs can cause BSODs then be very careful about who you ship it to.

Share Improve this answer

Follow

answered Sep 30, 2008 at 8:16

Windows programmer

8,055 ● 1 ● 24 ● 23

0

If it doesn't break anything else, why not ship it? It sounds like you have a good relationship with your customers, so those who want the feature will be happy to get it even if it's not all the way there, and those who don't want it won't care. Plus you'll get lots of feedback to improve it in the next release!

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:34

Mike Ivanov

153 ● 6

0

The important question you need to answer is if your feature will solve a real business need given the design you've come up with. Then it's only a matter of making the implementation match the design - making the "bugs" being non-bugs by defining them as not part of the

intended behaviour of the feature (which should be covered by the design).

This boils down to a very real choice of paths: is a bug something that doesn't work properly, that wasn't part of the intended behaviour and design? Or is it a bug only if if doesn't work in accordance to the intended behaviour?

I am a firm believer in the latter; bugs are the things that do not work the way they were intended to work. The implementation should capture the design, that should capture the business need. If the implementation is used to address a different business need that wasn't covered by the design, it is the design that is at fault, not the implementation; thus it is not a bug.

The former attitude is by far the most common amongst programmers in my experience. It is also the way the user views software issues. From a software development perspective, however, it is not a good idea to adopt this view, because it leads you to fix bugs that are not bugs, but design flaws, instead of redesigning the solution to the business need.

Share Improve this answer

Follow

answered Sep 17, 2008 at 22:44

Ludvig A. Norin

5,325 ● 7 ● 33 ● 34

Coming from someone who has to install buggy software for their users - don't ship it with that feature enabled.

0

It doesn't matter if you document it, the end users will forget about that bug the first time they hit it, and that bug will become critical to them not being able to do their job.

Share Improve this answer

Follow

answered Sep 30, 2008 at 8:54

Tubs

311 ● 1 ● 6