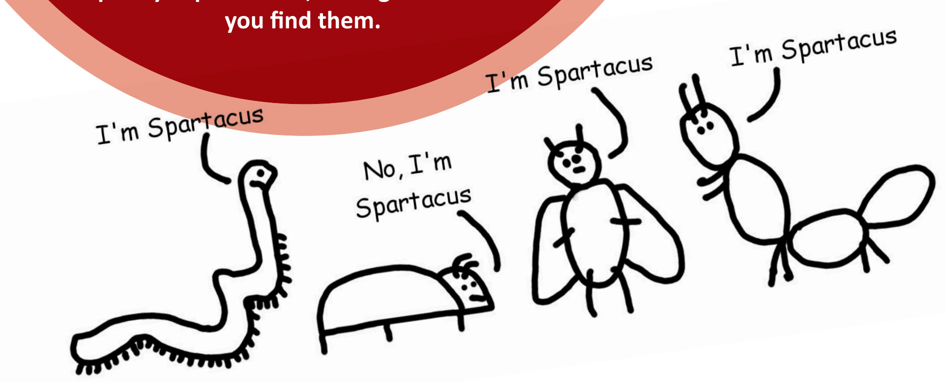
10 REASONS WHY YOU FIX BUGS AS SOON AS YOU FIND THEM

1. UNFIXED BUGS CAMOUFLAGE OTHER BUGS

How many times have you heard a tester say, “Good news, I’ve re-tested the bug you fixed and it’s working perfectly, but I’m now observing a new bug”? You might be in luck, fixing a bug may reveal no further problems, but postponing this kind of discovery is a risky strategy. What happens if that Priority 3 bug you’ve been ignoring has been camouflaging a Priority 1, or worse still, a bug that will require significant alterations to the software to fix? If you have one of these bugs hiding somewhere in your code, the sooner you discover it, the better. Don’t delay finding important problems; fix bugs as soon as you find them.

2. UNFIXED BUGS SUGGEST QUALITY ISN’T IMPORTANT

We’re all professionals at heart, but it’s surprising how quickly a team can find themselves in a downward spiral. A developer working on software that already contains hastily written, error prone functions, with little or no unit test coverage, is likely to add more code of the same nature. Similarly, a tester who has seen tens of reported bugs go unfixed is unlikely to be enthusiastic about reporting many more. Of course, it isn’t just developers and testers that are affected. Over time, every member of the team will start to ask themselves, “what’s the point”, why aim for a high quality product when a substandard one is the accepted status quo. Don’t set substandard quality expectations; fix bugs as soon as you find them.



3. DISCUSSING UNFIXED BUGS IS A WASTE OF TIME

Regardless of whether it’s part of project planning, during a dedicated triage meeting or just gathered around a desk, discussing unfixed bugs is a waste of time. There really is only one question that needs answering, “does this bug need to be fixed”? Everything else, whilst interesting, is just noise. Should we categorise this bug as Priority 3 or a Priority 4? How long will it take to fix? Should we fix bug 113 first or bug 114? These are all questions that could be avoided (including the often lengthy conversations that follow) if teams fixed bugs as soon as they found them. Without doubt, every project will have its share of bugs that need special attention, but few projects require this level of attention to be the norm. Don’t waste time with unnecessary discussions; fix bugs as soon as you find them.

What are you still doing on that chart? We were fixed last week!
Nobody told me!
No. of Bugs
Time

4. UNFIXED BUGS LEAD TO DUPLICATE EFFORT

The greater the number of unfixed bugs, the harder it is to identify whether a bug has already been reported. Imagine a scenario where there are only 5 unfixed bugs. When a “new” bug is discovered, it’s easy to identify whether that bug has been reported by someone else. Now imagine trying to perform the same task when there are 50 unfixed bugs in circulation. It’s either going to take a disagreeably long amount of time (time which could be better spent looking for other bugs) or the thought of such an overwhelming task will cause it to be abandoned, often leading to duplicate bugs being reported. These duplicate bugs lead to duplicate investigation and duplicate re-testing. Don’t waste time on unnecessary duplication; fix bugs as soon as you find them.

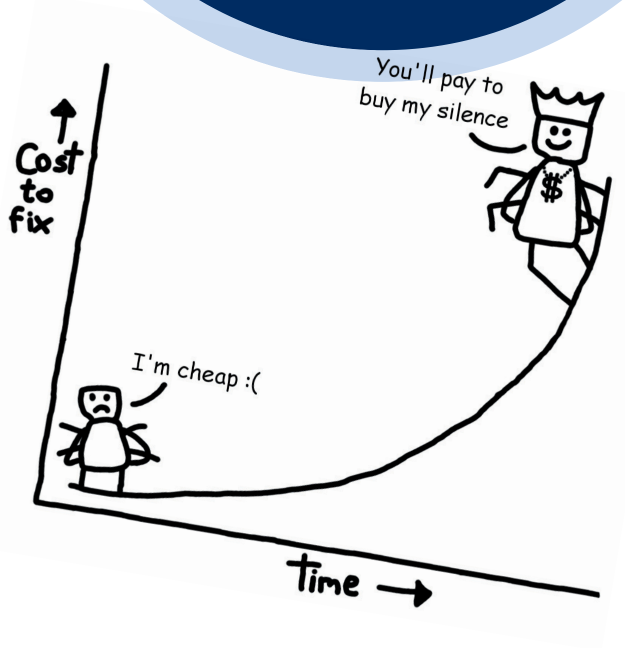
5. UNFIXED BUGS LEAD TO UNRELIABLE METRICS

Different teams analyse bugs in different ways. Some casually monitor how many are left to be fixed, whilst others track everything from their density to their lifespan. Regardless of the complexity, every bug-based metric relies on accurate underlying data. As the number of unfixed bugs increases, it becomes increasing difficult to maintain accurate bug information. Even if the information was correct at the time, the longer a bug is left unfixed, the greater the chance that information will diverge from reality. The resulting misinformation then ripples through the team. Don’t fall foul of project decisions based on incorrect information; fix bugs as soon as you find them.

There's nothing to worry about. We "just" need to fix a couple of bugs and then we're done!

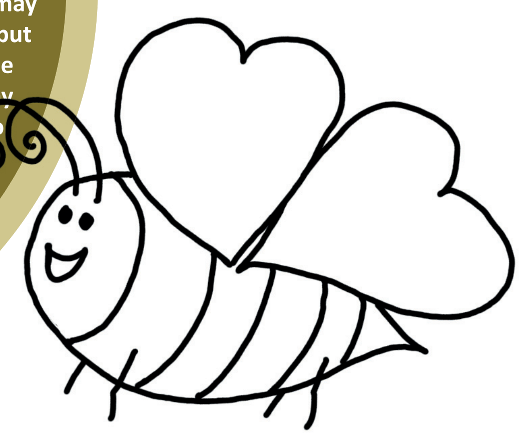
6. UNFIXED BUGS DISTRACT THE ENTIRE TEAM

When somebody encounters an unfixed bug a number of distracting questions are planted in their mind. Take a developer who is about to make an enhancement when they notice a bug. Should they fix the bug first, has somebody else fixed it but not checked-in, can they rely on the buggy code to be the basis for their own? Similarly, imagine a tester who has stumbled across a bug in one functional area whilst setting up the pre-conditions to test another. Should they postpone testing the intended area and instead explore around the bug they stumbled across, has this bug already been reported and would exploring it be a waste of time, could this bug (positively or negatively) pollute the results of the planned tests? Don’t let your team be distracted by unfixed bugs; fix bugs as soon as you find them.



7. UNFIXED BUGS HINDER SHORT-NOTICE RELEASES

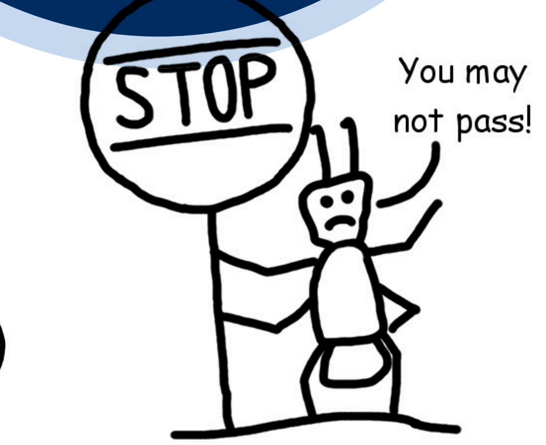
Once in a while an event occurs that forces a team to release all or part of their software when they least expect it. Maybe an emergency patch is needed to fix a bug in the production environment, or an unexpected visit from the project sponsor requires the latest release to be installed on a demo laptop. These events can be taxing at the best of times, often made worse by the presence of one or more unfixed bugs. It may only take a relatively short time to perform the release itself, but with unfixed bugs in the code, how long will it take to get the software ready for release? Even if a team can quickly fix any bugs blocking the release, there is also the time required to re-test the bugs to consider. The result is often a delayed release or a release that contains only the most glaring bugs removed. Don’t let your releases be hindered by unfixed bugs; fix bugs as soon as you find them.



"Look at me! Aren't I a pretty little bug?!"

8. UNFIXED BUGS LEAD TO INACCURATE ESTIMATES

No two bugs are ever the same. Some require mere seconds to investigate; others take hours to diagnose. Some take minutes to fix; others take several days. Some can be automatically re-tested; others require manual verification. Combine together these uncertainties and you can see why the more unfixed bugs a project has the less accurate their estimates become. It’s easy to fall into to trap of thinking that the effort required to fix and re-test bugs fades into insignificance compared to other project work and they can be ignored / managed via a healthy chunk of contingency – this is rarely the case. Even with a contingency in place and detailed analysis of each bug performed to understand whether the contingency is sufficient, a team can never truly know how long it will take to fix and re-test each bug until the work is complete. Don’t misinform your stakeholders with inaccurate estimates; fix bugs as soon as you find them.



9. FIXING FAMILIAR CODE IS EASIER THAN UNFAMILIAR CODE

The human mind is capable of many incredible feats, but retaining information indefinitely is not one of them. Over time our memory decays and things we used to know intimately become blurred and unfamiliar. The code a team writes is no exception and for this reason it is easier for a team to fix a bug in code they edited earlier that day compared to code they haven’t seen for a week or two. A team can reduce the effect of memory decay by sticking to good development principles, but this will only reduce the effect of memory decay, it can never alleviate it completely. Avoid the frustration caused by having to familiarise yourself with a piece of code you once knew; fix bugs as soon as you find them.

10. FIXING A BUG TODAY COSTS LESS THAN TOMORROW

For all the reasons listed in points 1 to 9, fixing a bug today will cost you less than fixing the same bug tomorrow. If a bug is left to fester in the software you are developing, configuring or maintaining it may camouflage other bugs, demotivate the team by suggesting quality isn’t important, become the topic of pointless conversations, cause duplicate effort, lead to incorrect project metrics, distract the project team, hinder short-notice releases, invalidate estimates and lead to unnecessary frustration. And the longer you leave a bug before fixing it, the more likely these things are to occur and to a greater extent. Don’t put your project at risk; fix bugs as soon as you find them.