There Are No Bugs, Only Problems

An Essay by Blue-Maned_Hawk

(HTML version | Source)

Many pieces of software have two methods of submitting feedback on the software: bug reports, and feature requests. In this essay, i will argue that the distinction between these is arbitrary and unhelpful, that having a dedicated space for feature requests encourages poor maintainance of software, and that the only thing worth keeping track of in a project is problems with the software.

What is the distinction between a bug report and a feature request? An average definition would probably be something like "feature requests are for requesting new things to be added, and bug reports are for reporting problems". This already leaves us with a problem: there's nowhere to request changes or deletions. Let's ignore that problem for now, and instead ask: Okay, so what counts as a bug?

An average definition of a bug would probably be something like "something that the maintainers of the software didn't intend". Notice that this definition does not implicate that there is something *wrong* with the unintended thing; this already leads to the contradiction of this definition with the common sense of "bugs are things to be fixed". Applying this bit of common sense indiscriminately would mean that things would get removed from the software *even if they happen* to be good. While this pitfall is *often* recognized by the maintainers of software, it isn't always.

A bigger problem with this definition is that it is vague. Suppose, for example, that some new object is added to a multiplayer game that ends up throwing off the balance of the game. Would a request to remove that object be more suited as a bug report, or a feature request? Either side could be argued here. Perhaps it should be a feature request; after all, the addition of the object was completely intentional. Perhaps it should be a bug report; after all, the maintainers certainly didn't mean to throw off the balance of the game.

In this particular instance, one will find that most people will intuitively think that it would be more appropriate as a feature request. Repeated trials will end up betraying a certain criterion: most people only consider something unintended to be a bug if it is some mistake in the *source of the software itself* (the code or the assets), not if it's a mistake in the design of something. Why? While it is true that there is no obvious benefit to this, closer inspection will reveal that there is no non-obvious benefit either. This distinction is arbitrary and unhelpful.

Now let's get back to that problem we ignored before: an average definition of "feature request" would probably not make any provisions for requesting changes or deletions. This is something that flies right in the face of intelligent software design: changing things that exist and removing things that are harmful are important tools in the solution of problems when maintaining software. By having a dedicated space for feature requests for a piece of software, users of that software are encouraged to come up with solutions to the problems that they encounter in the software that add new things to the software, instead of coming up with solutions that change what already exists or remove things from the software. These additions will result in the software becoming bloated over time—after all, software bloat is quite literally the addition of unnecessary things to software over time.

When maintaining a piece of software, what is worth doing? What purpose is there in doing something that does not fix a problem with the software? Even if something that the users of the software want in the software feels like it doesn't fix a problem, a deeper analysis of the request and an introspection into the desires of the users will often reveal that, unwittingly, they have requested it because there is some sort of *problem* in the software that needs to be fixed. But that solution, that request, may not necessarily be the *best* solution to the problem; often, brainstorming and experimentation is necessary to determine what is the most optimal solution to the problem.

By this measure, why should the solutions be the thing to keep track of? What matters are the problems that the solutions solve, so it should be *those* that are kept track of, with the solutions kept track of under the scope of the corresponding problems.

But problems do not fall cleanly into either bugs or features to be requested when we use the average definitions. Some would be better filed as bug reports; others would be better filed as feature requests; still others don't really work as either one, or teeter on the edge of one or the other. This makes the distinction between bug reports and feature requests unhelpful.

By this reasoning, a piece of software ought not to have separate spaces for bug reports and feature request. Instead, a wisely-maintained piece of sofware ought to have a single space for the reporting of all *problems*, where solutions can be discussed and experimented with. By this design, the software shall fight bloat and shall provide a more effective and useful way for the general user to contribute to its improvement.
