

I used to wonder why I see so many more bugs than most people. I easily observe [hundreds to thousands of bugs per week](#) and [nothing seems to work](#), but most people I talk to don't see [anything like this](#). For a long time, I thought this had something to do with how I use computers but, over time, I've realized that it's mostly that people are hitting the same bugs and don't notice.

If you're not a programmer, that's probably ~~a better way to see the world~~, but I think curing quality/bug blindness is helpful for programmers. I've done this with a lot of friends and acquaintances (just by pointing out bugs). After a few weeks, people who are so inclined tend to start noticing bugs as well.

Because I notice these kinds of things, I've had multiple jobs where directors/VPs/execs/etc. sometimes ask me to evaluate something when they want an actual opinion from someone who is relatively likely to notice issues (and fix them or drive fixes for them if necessary). Sometimes I won't find any issues (there are likely issues that just aren't the kind I notice). More often, I find issues that fall somewhere from "mild" to "moderate". And, sometimes, the issues are severe, to the point where one might even say the thing actually doesn't work.

I find this last category a bit mysterious, as when I look up discussions on how the thing got into this state, there's usually a stream of internal comments indicating that the thing is great, it works well, etc., but when I open up the thing and try it, it's in a state where the thing only works if you do quite a few non-intuitive workarounds. More likely than not, not only would a normal user not be able to use the thing, they'd have such a ~~hilariously/infuriatingly~~ bad experience that they'd tell their friends.

I've had this post in mind for maybe a decade or so, but I was hesitant to write it up because, in the back of my mind, I always wondered if I'm somehow triggering weird corner case behavior most users don't hit without realizing it. But after seeing more and more cases where the product launches and falls flat on its face because users run into the exact same issues I saw, I don't think that, in general, I'm hitting bugs because I'm doing unusual things a normal user wouldn't do. If a product seems severely flawed when I use it, it probably is. And with the magic of LLMs, nowadays, I can even have LLMs act like normal users in a lot of ways and show that the issues reproduce across many different scenarios.

A few examples

I don't want to give any specific examples where it was my job to see how well the thing worked because, even if the internal examples are meant in a constructive, blameless, way, they may not always read that way when re-posted externally, so I'll give a few less interesting and less well supported "random" examples.

A while ago, I [wrote up the results of some web search queries](#) and found poor results from Google, Bing and Kagi. In general, the major search engines failed to return good results for the queries and returned pages full of low-quality SEO spam as well as some sites that were actually scams. BTW, on the scale mentioned above, I would consider this "moderate" and not "severe" (~~severe would be something like, the search engine returns 500 errors half the time, the majority of results are scams, etc; my bar for severe is that a normal user likely won't be able to use the thing at all, not that they have a bad experience~~). Almost nobody¹ objected to my characterization of Google and Bing search results, but people told me that I was wrong about Kagi. In some cases, people sent me their actual search results. In every such case, the search results did not contain a good result that I could see (e.g., for the seasonal forecast query, the search failed to return an up-to-date seasonal forecast) and was full of SEO spam. In one case, a person passed me both their list of Kagi filters as well as the search results they got without making claims that the results were good or bad, but people generally insisted the results were good even though the results both failed to link to a useful result and were

full of spam except in cases where the user did something like pin GitHub to the top of their results, which worked for the queries where the goal was to download software that's hosted on GitHub, but of course completely fails for the other queries from the post.

In the abstract, I get that people who are fans of things tend to be blind to the thing's faults. For example, since I bought a Volvo after [seeing how they do in out-of-sample crash tests](#), I sometimes search for answers to my questions on Volvo car forums. For well over a decade, the reliability data that exists (and I think this is backed up by the anecdotal experience that mechanics who work on Volvos have) is that Volvo reliability is mediocre to poor, but of course Volvo forums are full of people who insist that Volvos are among the most reliable cars and that the data are all wrong.

An example that might be more central to the topic is Blackboard (the course management software). Back when it was the most widely used software by universities for coursework, the software was widely disliked by both students and professors. I think it would be fair to say that it was the most widely disliked software in my social circles (there was more strongly disliked software, like Visual Source Safe, but any more strongly disliked software wasn't widely used enough to be the most widely disliked overall). The Wikipedia page notes

Blackboard had become "one of the most disliked — even detested — companies in education."

as well as

In December 2011, Fast Company reported that 93% of respondents to the Amplicate customer opinion survey "hate" the company.

Back when I was much younger and had less of a filter, I ran into someone who worked at Blackboard and, without thinking, I stupidly blurted out something like "what's it like to work on this software that so many people dislike?". Luckily, the person I was talking to wasn't offended at all and, instead, they were actually confused because they thought it was widely loved software that users really liked. They didn't really believe what I said could be true and I made some comment indicating that it was just confusion on my part and then the conversation continued in a different direction. At the time, as someone much younger and more naive, I was really surprised to hear that the software that was probably the most widely disliked software in my social circles was thought to be really well-liked software by the one employee from the company I met (and, presumably other employees as well).

I can understand how the Volvo forums get to be how they are, in that [cars are reliable enough in general now that people generally don't experience car breakdowns](#), so it's easy for someone to think something like "the data can't be right; after all, my car has never broken down". It's more of a mystery to me how somebody can look at a set of search results that are full of spam and then dash off a message explaining how great the results are, even if they're a fan of a particular search engine or how someone can think that users generally love software that's famous for being disliked, to the point that every single person I talk to about it tells me how bad it is (often in unprompted complaints), there are news articles that discuss how much people dislike the software, and the near-universal dislike for the software is mentioned on its Wikipedia page. Another Blackboard-like example might be Discourse (forum software) web performance, where one of the inspirations [for this post](#) was discussions with Discourse employees who thought that Discourse had great performance. I found that one interesting because Discourse actually had code in it that slowed down actual page loads [in order to cheat on web performance metrics like LCP](#). That went well beyond [just optimizing for a benchmark](#) and rose to the level of actual cheating that not only had no benefit to the user, it actually harmed the user. At some level, the programmers implementing that sort of cheating and advising users on how to not accidentally subvert the cheating must know that the actual performance of their app is poor, but it's very easy for people to put up mental barriers around this kind of thing.

By now, I wouldn't say that I'm surprised because I've seen this kind of thing enough that I would actually consider it surprising if it didn't happen, but I still wonder what's going on inside someone's head when something like this happens.

For a non-programming example, we previously noted [in this post on how people have different perspectives on "obvious" facts](#), there's a basketball player who, subjectively, is generally considered to be the dirtiest player of his era. The NBA doesn't track objective measures of player dirtiness, but he seems dominant on a wide variety of measures. For example, although, like rebounds before 1950, genital strikes aren't an officially tracked stat, he surely holds the record for punching, kicking, kneeing, or otherwise striking players in the genitals this century (he should also hold the record for era-adjusted numbers, but it's possible that he doesn't have the all-time record due to play being much dirtier overall in the 80s and 90s). In discussions, most fans of his team don't seem to notice this and the phrases "natural rebounding motion" and "natural shooting motion" have become running jokes from how oblivious the team's fans are when they justify this player's contortions when he strikes other players in the genitals.

On average, humans have a high ability to ignore negatives in things they're a fan of, including (and often especially) their own work or work their company does. For better or for worse, I seem to have the opposite of this and my thoughts immediately go to the flaws in myself and my work. A number of times, as a result of a blog post, someone has messaged me with something like "how would you like it if someone criticized your work?" or "how would you like it if someone said your work isn't good?" To the former, my thought is that I go to great lengths to get criticism from people who can poke holes in my reasoning, so it's pretty awesome if someone has remotely reasonable criticism of my work. And to the latter, I generally think my work is full of major flaws, so, uhh, yeah, it seems pretty reasonable to say it isn't good. There are particular aspects of my work that I think are interesting or good but, overall, I don't know that I'd rate anything I've done as good. ~~I'm not saying I don't have blind spots~~, but I think I'm a bit less prone to this particular one than most people².

Habitual mitigations

If I think about analogous blind spots I've had, one that jumps out at me is from when I was a little kid and a friend of mine used my computer. For this story to make sense, you have to know that this was in the [mechanical mouse](#) era. Over time, detritus would get stuck to your mouse ball and cause it to track erratically unless you cleaned it out.

When my friend tried to use my computer he found it impossible to use the mouse because mouse pointer movement seemed almost random. When I sat down at the computer again and used the mouse I didn't have any problem using it at all, but on looking at what I was doing with my hand to smoothly move the pointer in a straight line, I was violently throwing my hand all over the place. I realized I must've adjusted to the detritus on the mouse ball over time as it accumulated and I was somehow compensating for the mouse's extremely erratic tracking by making countervailing erratic movements³. I thought it was pretty amazing that I could not notice that I was doing this and I always wonder if I'm doing some equivalent thing today.

I sometimes think about all of the mitigations I've developed to work around bugs. For example, when opening a new Google Doc, I used to immediately put the title I wanted into the doc. At some point, maybe ten years ago or so, Google Docs added some kind of delay such that the typing you do into the title box right after you open the doc gets overwritten, so I now have this habit where, after opening a Google Doc, I do something else and then I change the title. Over time, as Google Docs has had more and more features added, I've developed a series of habits that avoid all sorts of pitfalls (such as trying to search at the "wrong" time and getting the useless native browser search instead of the Google Docs search).

My feeling is that a large fraction of computer literacy and software literacy is developing a large library of these habits that you just do at a non-conscious level. These are often quite specific to the situation, such as

a habit I developed when I worked at Microsoft of flipping my laptop's WiFi switch to off before logging in (which I noticed other people doing as well). This was because there was some service, which would often fail your login with "There are currently no logon servers available to service the logon request". But if that service couldn't connect at all, the check would be bypassed and you could just log in.

Quality blindness

We could fill a post up with examples like that, but back to the main topic of the post, one commonly suggested way to try to overcome quality blindness is to have people dogfood their own software. On average, this is a lot better than not dogfooding, but it only works to the extent that people don't figure out (and then forget about) habits that work around whatever issues the software has. On average, programmers are pretty good at working around software foibles (you had to be in order to be an effective programmer pre-LLM), so it's very easy for programmers to not notice these kinds of issues if they're not paying attention.

On the flip side, a large part of making an app easy for people to use seems to mean making weird habits like these unnecessary. Although this sounds like it should be easy to do, from having seen people try to give feedback about this kind of thing, the reflexive reaction of most developers seems to be "huh? It's easy to do X, just do [complex sequence of things that no normal person would think of if they hadn't used the app many times before unless it was specifically explained to them or they saw someone else do it]" or "huh? Didn't you see that the instructions for this are clearly laid out in page 43 of the manual after you execute the steps in Appendix B on page 261?".

That being said, I think curing people of quality blindness is do-able because I've done it quite a few times. I think this only really works when the person is receptive, as people have infinite capacity for willful blindness but, in cases where people are receptive, just pointing out issues they didn't notice seems to work. Years or even a decade later, people will sometimes tell me they see bugs everywhere now.

The reason I think this is worth doing is that I've seen people and teams with a high degree of quality blindness ship things that have reduced or even no chance of success because of product quality issues⁴. It's one thing to knowingly and deliberately trade off quality for speed⁵, but when I've seen this happen there's always been a kind of quality blindness where everyone involved with the project thinks they're shipping something very high quality when that's not the case.

This has never been unimportant, but it's gotten more important with coding agents because, while it's easier than ever to churn out low quality software, it's also easier than ever to improve quality, whether that's [better performance](#), [fewer bugs](#), etc.

But, to do this, you have to [actually notice](#) that this is possible, that [quality can be improved](#).

Thanks to Yossi Kreinin, Dennis Snell, Michael Malis, Emu Chu, Gary Bernhardt, Jon Surrell, and Matt Mullenweg for comments/corrections/discussion.

Naturally, Gary Bernhardt ran into a Google Docs bug while reading a draft of this post.

P.S. Like I've mentioned in the last four posts, I've been trying to write posts more quickly because, with LLMs, it's so much easier to look at data and figure things out but, since I'm not writing with LLMs, the time it takes to write something up hasn't fundamentally changed, unless I want to move to a different point in the quality-velocity trade-off space. The prior result was that I would run some experiments and tell a few friends and then never write anything up because, due to Amdahl's law, writing anything up would effectively consume all of my bandwidth for running experiments. In fact, despite trying to do this (my goal is to spend 30 minutes per post on the write-up), since writing my last post, I have three results that I think could make a totally fine blog post

that I haven't had time to write up (not including things done for work, which would add a few more things). Without having LLMs write for me, I don't see a reasonable way to get the time per post significantly below 30 minutes (and I think I often miss my goal and take more than 30 minutes), so the non-LLM options here are some posts that are much sloppier than my normal posts (in a human slop kind of way), or almost no posts.

Anyway, if you have opinions on these quick (and surely more wrong) writeups, let me know what you think ([X Bsky Mastodon](#))!

Appendix: advertising blindness

Michael Malis (founder and former CEO of [Freshpaint](#)) noted (in messages, hence the message-like format)

For a similar but different data point - I've seen similar blindness when it comes to advertising. When I would explain Freshpaint to people, I would tell them that we help hospitals with marketing

A common question I get is why do hospitals do marketing. The weird thing is if you pay attention, hospitals do a ton of marketing

In SF there's tons of bus ads and billboards for ucsf/sutter health/stanford and various treatments

This is a different topic from both Michael's comments and the post, but I'll say that I've talked to quite a few people who don't believe ads work at all, but I talked to someone whose data methodology and judgement I trust about ads A/B testing at one big company I worked for and looked at the data myself at another company and I thought the causal evidence for ads providing real lift (well beyond the cost of the ad) was strong in those cases. In the case where I looked at it, they did a geo-segmented A/B test where they bought ads in some geos but not others (this was done worldwide, with the regions being things like U.S. states, Canadian provinces, etc.). This kind of geo-segmentation was done because, even with cross-device tracking, it's not 100% clear if someone has been exposed to an ad or not (of course this is still the case with this kind of segmentation and I would prefer segmentation that was more clustered to population areas and didn't have splits where people are relatively likely to, for example, commute from one side of a boundary to the other, but this kind of contamination generally makes the likely true lift higher than the estimated lift), so people sometimes do these geo-segmented A/B tests.

Anyway, in these A/B tests, return on ad spend was quite good just on direct revenue gain, and there was also a gain in users which seems likely to result in more revenue down the road (the later revenue wasn't analyzed). I don't know about ad effectiveness in general or if your particular ads are effective, but the [commonly repeated idea](#) that ads don't work in general seems wrong to me.

On the topic of Michael's comment, I think it's easy for programmers to not notice ads. Almost all programmers I know use an ad blocker and, in real life, their eyes seem to just skim over ads and not notice them. I can see how this would feed into the idea that ads don't work. Who the heck would look at these things? But from my interactions with "normal" people as well as the data I'm familiar with from my time at Google, many or perhaps most people don't even realize that a lot of ads are ads. When they do a Google search and they click on the top result, they often have no idea they're not looking at what Google "thinks" is the best link, they're looking at a link from whoever paid Google the most to buy that ad slot.

Appendix: comments from other folks

Em Chu, on a habitual bug mitigation:

I'm sure you can collect infinite examples for this section, but I just want to Complain: when waking up and unlocking my laptop (mac), it's very easy to get it in a state where it's "awake" but unusable

(black screen with cursor or similar) which can only be fixed by physically closing the lid and re-opening it. To work around this, I think I usually wait a second after the screen turns on, interact with the trackpad, and then unlock it, though honestly that happens mostly subconsciously, and I clearly need practice given that I still hit the bug a few times a month.

On reading this, I examined how I open my laptop and realized that I have some funny habits as a result of working around other laptop bugs. The specific bug mentioned here doesn't reproduce on my laptop and it seems that I can stop the habitual mitigation I put into place for some prior laptop.

Gary Bernhardt, on his experience reading a draft of this post

While reading it, Google Docs' UI seems to have broken, making it impossible to scroll up to read some comments (see screenshot [not shown in post]).

From looking at the screenshot, I've seen the exact same bug and have some mitigations for it (different ones depending on the context). I would personally rate Google Docs as far above average in terms of software quality: I find it much less buggy and janky than the major alternatives (Microsoft Word, Open Office, various old editors that are long gone like StarOffice, Lotus, etc.). And yet, I could easily sit down and write a 10k word post on Google Docs bugs and the workarounds I have for them.

At times, I've tried to see if I can get a job somewhere where I just fix quality issues all day. This has never panned out, due to some combination of this not being a very high priority and it also not being a normal role that companies have a role for. I sometimes daydream about joining companies as an intern and just fixing quality issues for a few months and then leaving. In practice, I think if I got such a job, a lot of the fixes would get blocked and it would be very difficult to actually drive change as an intern for three months, so it would have to be some mostly abandoned project where nobody cares what I do (and corporate priorities aren't so focused on shipping features that fixes get immediately re-broken).

[@IncidentNoodle](#)

Unintentionally on topic: the `<abbr>` tags worked on mobile ~last week, but are no longer working across any iOS browser (safari/chrome/firefox), and I had a hard time figuring out that hover showed them on macOS browsers (all three) due to the long delay

[@gunchleoc@mastodon.scot:](#)

Germans have a word for that - [Betriebsblindheit](#)

[@oulipien.bsky.social:](#)

Crazy anecdote from @danluu.com here and I wish he'd been even blunter at the time and asked this person where they'd gotten this belief about Blackboard being liked by anyone at all. User surveys? Principal (as in, not agent) surveys? Inner conviction??? [screenshot of Blackboard anecdote]

[Some variant of, people are forced to say that they don't see bugs by their bosses]

I don't think this is consistent with any of the major examples in the post, let alone all of them. Consider the Blackboard example mentioned above. It's unlikely that I and other people this Blackboard employee are "secret shoppers" who are checking in on employees, and the employee's reaction is clearly absurd to anyone who isn't such a hypothetical (and in reality, non-existent) secret shopper, so reacting like this just makes them look a bit silly in the eyes of a large fraction of the people they meet for no benefit (for example, see the previous quote, which seems like a typical internal reaction). Perhaps a few very paranoid employees would

maintain this front on the off chance they run into some friend or relative of the boss who knows that they work for the company who relays the story back and they have a boss who would care about this, but it's just not plausible that this is (for example) the case for every Discourse employee who reached out to me to explain to me that Discourse performance is really good.

If we look at the basketball example, this is even more absurd. You could possibly come up with some kind of reasoning like, other fans would shun you if you didn't believe or pretend to believe the most absurd rationalization, but as someone who has spent a lot of time around sports fans, I've generally not found this to be the case. And, to the minimal extent to which this is kinda sorta the case, it's more an issue of self selection, where fans who are into the most extreme rationalizations will spend more time around fans who are into the most extreme rationalizations and fans who are less into these extreme rationalizations will tend to spend more time with fans who are less into them.

Also, just looking at the career path of people who don't buy into these things and fix them, noticing these issues and fixing them has gone very well for those people. Pretending these issues don't exist (whether that's at a conscious level or not) also seems to work well, so I don't know that fixing these issues is actually a better career path, but it's certainly not so bad that, in general, there's meaningful career pressure to pretend these things don't exist overall even if there are some individual positions where there's some direct pressure to pretend these issues aren't real.

[Daniel Gibson](#):

Who else uses the Shift key to end the screensaver, because in case the event goes through to an actual program it's least likely to do have unintended effects?

This reminds me of how, when I want to send a queued message to codex immediately and interrupt the current tool call, I put my finger on the key and the press as quickly as possible to reduce the window of time where the tool call will finish and the escape key will stop codex entirely instead of causing the message to send. I should probably just run a patched version of codex that has fixes for this and a few other issues I've run into, but I'm already doing things like [trying out some weird workload-specific optimized version of ripgrep](#) that also has an added native code compiler which compiles matching expressions in another thread while the search starts and then cuts over after compilation completes, so it's not like I'm against creating weird patches to improve my workflow and it's more of an issue of overall bandwidth (no doubt, on writing this, someone will tell me that I could just hit another key instead and could've found this out by asking codex about the key in the time it took me to write this comment). Just like with Google Docs, I consider codex above average in terms of software quality in the space, but even though I haven't been using it for a year, I could easily write 10k words on all the workarounds I've implemented (either by habit or, in some cases, with actual scripts that monitor for broken behavior and then correct it).

[John Regehr](#):

I didn't remember this until after I'd written and published this post and someone linked to John's post, but apparently John Regehr wrote "Operant Conditioning by Software Bugs" before I started a blog! Maybe I should've used an LLM to search for prior art, but if I did that, I'd probably never write anything because there aren't that many really new ideas and almost everything is going to be similar to something someone else has said. For better or for worse, I'm much more verbose than John, so this post uses a lot more words and has more random stories thrown in. If you find my blog posts too long, but have somehow managed to stumble down into the bottom of the post anyway, you'll probably like John's post more than mine :-).

-
1. someone told me the results didn't reproduce on Google when they tried it some number of weeks later. Of course it didn't, [which I discussed here](#) in more detail, but for the short of it, here's this post about

scams and other bad results on Google that was #1 on HN for a while. Of course somebody fixed that! And, also, ad results are non-deterministic and, while there are a lot of bad ads, it's not like the majority are scams, so you wouldn't expect to get scam ads at the top results even if someone else did for the same query. [\[return\]](#)

2. For example, anyone familiar with my code at Twitter will recall the huge comments I had at the top of the main files for the things I owned, which described the various ways in which the thing is really flawed. They were all things that, for one reason or another, I thought weren't worth the time to fix, but they were still serious problems that anyone interacting with the code ought to know about. For [this metrics project](#), I even had a long doc that described the issues in great detail (IIRC, in a lot of cases, the rough shape of the fix was described; maybe today an LLM could take that and fix it).

I have the same feeling about my writing. While a huge number of bugs sneak through my writing (like spelling and grammatical errors), most of those are things I sort of don't care about and will skim past in other people's writing as well. When I say don't care, it's not that I don't want things to be better (when people send me corrections I generally fix things), it's just that my brain doesn't naturally pay attention to those things no matter whose writing it is, so I don't seem to have a particular blind spot in my writing with respect to these kinds of bugs. For the things I do care about, I could edit posts endlessly because, no matter how much I edit, the post still seems pretty bad to me.

I used to often (and still sometimes) send a post to someone and ask them if it makes any sense to publish it at all because I generally don't like my output and, if I'm just looking at my own writing, I don't think it's worth publishing. At this point, I've done this enough that I'll often just publish even though I don't like what I wrote, but if someone says "how would you like it if someone told you your work wasn't good?" as a kind of "gotcha", boy, they really have no idea how I think about my work.

There are various tricks I've used to get around this (not explicitly to get around this, but they do so as a side effect). As [discussed in this old post on writing](#), for a while, I hired a professional editor and had a process goal of doing one pass on each post and then trying to improve the next post. And as noted in the postscript to recent posts, now I'm trying to write with extremely minimal cleanup and editing and push posts out in half an hour regardless of the state of the data I'm looking at or the post (which I'm generally failing to do; I thought I might succeed on this one because it doesn't have any data analysis, but someone made a comment on the draft post that got me to re-write the whole thing, and just on number of words in the post, half an hour would really be pushing it on the original and then it increased in length). Of course a post that's written as quickly as possible with little to no regard for cleaning things up is going to be terrible in all kinds of ways, so all flaws I see in the post don't stop me from publishing it. Have my recent posts been good? Of course not; for any of the experimental/data posts, I could probably name ten things that should be fixed about each of them off the top of my head. For this post, I'd have to re-read it to come up with ten things, but I'm sure if I did re-read it I'd want to re-write the whole thing because of the issues it has.

[\[return\]](#)

3. This was, inadvertently, a kind of revenge on my friend for when I tried to open his door for the first time to leave his place. Since the door clearly opened to the outside, I tried pushing on the door, which didn't work, so I checked if there was a latch that was stuck, if the door was still locked, if I needed to push harder, etc., none of which worked. When he saw that I couldn't open the door I asked him what the trick was he said, in a tone of voice that made it sound like this was obviously something everyone should know, you need to pull the door before pushing it. The door was wedged such that the easiest way to open the door was to pull the door as tightly shut as possible and then immediately shove the door open.

This friend, since he grew up in that house, thought this was obvious, apparently not realizing that it's not normal to have to try to close a door extra hard to open it. [\[return\]](#)

4. a response I've heard to this kind of thing recently is that Anthropic had the best growth numbers in history while Claude was very buggy. If you have the best coding model and agent in the world, you can get away with a lot, but even they seem to have spent a fair amount of effort improving quality.

Maybe you can also get away with it if you have a product that succeeds due to bundling, the strength of your enterprise sales team, network effects, monopoly power, etc.; all but one of the cases I'm thinking of are places where the team didn't have these things on their side. I actually thought the one other case I was thinking of would be something like Blackboard, but (if the Google results are accurate) I see that the software has declined from being #1 in the market to being a minority player, so maybe they couldn't get away with it either (I didn't look into the reasons for the decline; perhaps it's a coincidence).

As noted above, Blackboard is an example where you could argue that the software quality didn't matter and people might as well just believe whatever makes them happy; if thinking that users love the software, then why not think that? But most of the rest of the examples that come to mind for me aren't cases like that. I don't think this is the best example, but it comes to mind because the comment below is the last time I was reminded of the Blackboard example. [There was a comment from a Tumblr employee who said that they'd solved the moderation \(abuse / spam / toxicity / etc.\) problem mechanically at Tumblr](#) via the way reblogs worked and that the mechanics Tumblr provided to users were good enough that the community could self-police bad behavior and that other social media sites would do well to learn from Tumblr. This was referring to Tumblr back in its heyday (maybe 2009-2014). I never really read much on Tumblr so I don't personally have an opinion, but back when it was a major social media platform, the reputation among folks I know was that it was heavy on bad behavior, particularly pile-ons caused by people taking out of context quotes and turning them into ragebait (not to say this doesn't happen on other platforms, but the belief was that the way Tumblr was structured and/or the communities involved made this worse on Tumblr). I'm not sure I know anyone who used Tumblr at the time who would say that the community was good at self-policing. In fact, when Scott Alexander wrote one of his most famous pieces, [Toxoplasma Of Rage](#), he dedicated an entire section to how Tumblr's reblog system is particularly bad and is guaranteed to result in bad behavior. He actually says that whoever designed the system either didn't understand what they were doing or they understood all too well and deliberately made the most ragebait-inducing system possible. This was written during the time when this employee said that Tumblr had solved the moderation problem and uses examples from that time.

Moderation at scale is an impossibly hard problem, so as a non-Tumblr user, I'm not even sure that Tumblr did worse than other platforms given its size and growth rate, but I think you'd need some quality blindness to think that Tumblr had solved the moderation problem. I think the strongest positive case you could plausibly make would be something like "Tumblr was better than average, but many people had a worse than average experience due to the communities they were in and some of these communities were unusually widely read and Tumblr therefore unfairly gained a reputation as being a particularly bad platform". I don't know if that's true or not, but it doesn't seem impossible that it could be true; it does seem impossible that Tumblr solved the moderation problem.

[\[return\]](#)

5. most of my projects are deliberately low quality; what I try to do is do the highest ROI testing, not test to the point the quality is what I would actually consider good, and this also goes for things like making interfaces very nice, etc. [\[return\]](#)

[There's no reason for software to be slow anymore →](#)

[Archive](#)

[Patreon](#)

[Mastodon](#)

[Bluesky](#)

[Twitter](#)

[RSS](#)