Recently my colleague tweeted an article about “defect density”, a metric used for showing the quality of the software under test. I found the article very interesting and thought provoking - in my experience, metrics like this can tend to make testing worse, since they focus on the amount of bugs produced rather than the quality. I once worked with a guy who, instead of logging one catch-all bug for a menu and sub-menus, he wrote about 16 bugs detailing for each menu screen that some text was 2 pixels off - simply because we were judged on the amount of bugs we logged.

Beware! Calculating defect density is totally worthless as nothing is ever tested until all the paths are exhausted <https://t.co/yrPUZPDlc5>

— Rosie Hamilton (@Rosicadia) [April 8, 2016](https://twitter.com/Rosicadia/status/718342536377606146)

**What Is Defect Density?**

Defect density is a metric that states “the more defects in the software, the lower the quality is”. This statement does make sense at first glance, since generally the more buggy the software the worse it tends to be. However there are so many different factors that lead to high or low bug counts that defect density is not a useful metric.

Defect Density is a key quality indicator. You can’t go wrong with collecting and presenting this defect metric. What’s more? It is one of the easiest to compute.

Hmmm...I’m not sure I necessarily agree with that.

**Why Defect Density Doesn't Define Quality**

Defect density doesn't take in to account any factors other than number of bugs per area of software or per number of lines of code, but there are many factors that can affect the quality and the number of defects found in a piece of software. Below I've listed a selection of factors that can affect the defect density and why I think defect density is not a good metric for defining the quality of the software at hand.

**Tester Skill**

The first factor that could affect the defect density is how skilled the tester is. A highly skilled tester will be more likely to find more bugs (and bugs of higher quality) than a lower skilled tester. If a bug hasn’t been found yet does that mean that the quality of the software is better? Or is the software quality still the same, since the bugs are still in the system but not recorded anywhere? Defect density simply tells the team the amount of bugs found in the software, and nothing to do with actual quality. No defects found != good quality, it could simply mean unskilled testers have been employed.

**Time Spent Testing**

Defect density doesn’t take into account the amount of time spent testing. It simply takes a snapshot of time and states “this is how many bugs are in the software for this area/lines of code at this time”. More time spent testing should yield a higher amount of bugs, just as having a more skilled tester will. And once again, does more time spent testing mean the quality of the software is less? Surely the more bugs found, the more bugs can be fixed, and afterwards the overall quality will be higher? Defect density once again doesn’t give us any useful information, besides the obvious “there are bugs, and some should probably be fixed”.

**Defect Type**

Something that defect density doesn't take in to account is different bug types: trivial, minor, major, critical and blocker. If I have a product that has four minor defects, versus a product with four major defects, which product has the lower quality? According to defect density, both are the same! Some products end up with hundreds of trivial and minor bugs - normally simple UI issues that no one really cares to fix immediately. The bugs that are generally important and telling of the quality of the software are the major, critical and blocker bugs. Defect density doesn’t include these in the calcalutions per line of code or area, and it's a pretty critical piece of information to be lacking from any sort of quality assessment.

**Usability**

Defect density doesn’t give any information on what the user experience of the software is like. Defect density doesn't tell me if navigating the software using a keyboard is a nightmare, or if moving back and forth between webpages is clunky and annoying. Most of the time usability issues like these are reported as "improvements" or "suggestions" and don't end up in the bug count, though these issues definitely do give some indication of the quality of the software.

**Temptations**

One of the issues with using a metric like defect density is the temptation to “game” the system; raising many similar bugs to increase the defect density OR bundling bugs together to make it seem like there are less bugs. There could even be the temptation to not report bugs if having a high defect density is seen as a bad reflection on the team. Just like with judging the quality of the tester on the amount of bugs raised (another outdated metric), the quality of the software shouldn’t be judged with the amount of bugs raised either.

**Conclusion**

I think that using a metric like the one described above is an outdated and risky practice. Instead of using metrics like this, testers should be able to communicate their opinion of quality. There are three questions that I ask when judging the quality of a piece of software:

1. Can you use it?
2. Does it meet customers requirements and needs?
3. Do I enjoy using it?

If I have issues with any of these questions, I communicate these to my fellow teammates and we discuss the surrounding issues. Quality tends to be a very subjective thing and I don't think it can be quantified. If you are happy with the software, your team is happy, and clients and customers are happy, then I believe you have a quality piece of software. No metrics needed.