Measuring Software Engineering

By

Oisín Thomas Morrin

18321152

(XXXX words)

**Abstract**

In this essay, I will open with identifying the scope of measuring software engineering, before detailing how software engineering can be measured, the platforms that can be used to gather and process the data, the algorithms that can be used to do it and finally answer the personal question of whether it is ethical at all to do any of this - and if it is, to what extent.

**Introduction**

The initial question of whether measurement is required can be easily answered with a resounding yes. Lazlo Bock of Google found that from a hiring perspective, the best indicator of the success of a future employee at the company was taken from a candidate performing a work sample test (29%) (Bock, 2015). But one could make the argument that perhaps basic success on the task is all that matters. Interestingly, the second best indicator is a general cognitive test (26%) (Bock, 2015), which would serve to make the point that, though binary success or failure is important, it is still too general, and that there is more at play. Working inductively from the hiring stage, it is evident that monitoring the work of employees would be of benefit. It is to make the general precise that the use of metrics is becoming pervasive.

But to what extent will we be considering these metrics? When considering measuring software engineers, there are a variety of metrics available to the organisation or person in charge of the engineers. The first question is whether the issue can be dealt with by analysing the individual alone. The measure of the productivity of the individual seems to be the keenest way to decide from a management standpoint whether to retain or fire and hire again.

If developer productivity can be measured, how is there not a single standard of measuring this in the industry? It tends to be because of a process called Goodhart’s Law: Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes (Wikipedia, 2020). In other words, all simple metrics will be gamed - be that lines of code written, number of commits, tickets posted etc.

What of the team? As has been seen in recent years, software engineering metrics for the individual can dissolve in the face of an awesome team: Whatsapp was a team of 50 engineers that now serve over 2 billion users (Bucher, 2020), and Instagram started as a team of 13.

So it is clear that a mixture of personal and interpersonal metrics will be needed.

**How can SE be measured?**

As a hands-on engineering manager, there a *lot* of things that you can measure to observe on the team. These go from the number of commits, average commit size, time to review, number and frequency of code reviews, time to merge pull requests to the main branch, and so on. None of these metrics are helpful by themselves, but together, they can paint a picture, and draw attention to places that could be problematic. And all of them only make sense in the context of the team.

There are a lot more things we can measure, though, at the team level. Like the output of the team. What has the team shipped? What was the business impact? How was the quality: how much rework was needed afterward? And, of course, let's not forget about the measurable things about the team. How is morale? What about attrition? How are people growing professionally: are they getting promoted?

CONTEXT:

For example, is one of the developers taking 3x as long to merge their changes, and do they get 3x as many code review comments as others? If they are a new joiner, this is expected. But if they are a veteran on the team, maybe there's something other going on there? Similarly, looking at these numbers can help identify top performers who are going over and beyond with code reviews, on top of their work. You can spot brewing conflicts by reading through pull requests with lots of back-and-forths.

There's a wealth of data to look at: but none of it is useful without context. Tools like [Velocity](https://codeclimate.com/) and [Pluralisght Flow](https://www.pluralsight.com/product/flow) expose much of this data: but they will never be able to deliver the context. For example, you might see that the number of commits and code reviews has dramatically dropped from an engineer. But when digging in, it might turn out they were pairing with team members, speeding them up, mentoring them, and adding long-term value outside the sprint. You won't be able to tell any of this until you look behind the curtain.

What platforms can be used to gather and process data?

<https://www.progression.fyi/>

What algorithms can we use?

<https://semmle.com/assets/papers/measuring-software-development.pdf>

Is this ethical?

**Conclusions**

**References**

Bock, L. (2015). Here's Google's Secret to Hiring the Best People. Wired. Retrieved from  <https://www.wired.com/2015/04/hire-like-google/> on the 13/11/2020.

Bucher, B. (2020) WhatsApp, WeChat and Facebook Messenger Apps – *Global usage of Messaging Apps, Penetration and Statistics*. Messenger People. Retrieved from

<https://www.messengerpeople.com/global-messenger-usage-statistics/> on the 13/11/20.

Wikipedia. (2020). Goodhart’s Law.Retrieved from <https://en.wikipedia.org/wiki/Goodhart%27s_law> on the 13/11/2020.