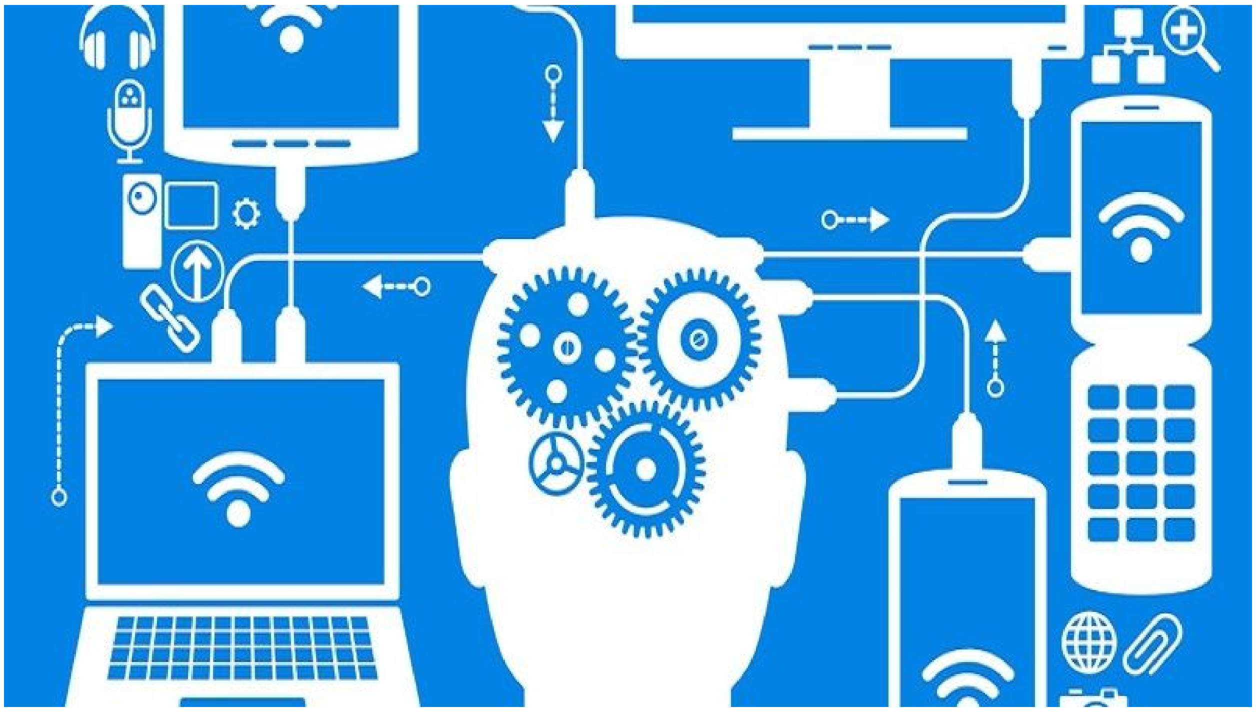
Measuring Engineering

CS3012



Shane Carmody, 15323753

Introduction

The task is to deliver a report that considers the ways in which the software engineering process can be measured and assessed in terms of measurable data, an overview of the computational platforms available to perform this work, the algorithmic approaches available, and the ethics concerns surrounding this kind of analytics. While there has been many attempts to measure the performance of a software engineer, it is still said that there has been no official way of doing so. Attempts are still being made as of today.

Measurable Data

- **Lines of code**
- **Number of Commits**
- **Keystrokes**
- **Test Coverage**
- **Speed**

Lines of code

In terms of a software engineer, one can think of many ways in which to measure them such as lines of code, although in this practice it's usually the less lines of code the better but obviously this can't be true for someone who writes 1 line as opposed to someone who writes 500 lines in the same time period. Although this practise shouldn't be used as one line of a high level language could take days to write and debug in assembly.

It really is a terrible metric but it does gives you an idea of the complexity of a system. If you're comparing two projects, X and Y, and X is 10,000 lines of code, and Y is 20,000, that doesn't tell you much - project Y could be excessively verbose, or X could be super-compressed. On the other hand, if one project is

10,000 lines of code, and the other is 1,000,000 lines,the second project is a lot more complex, in general. The problems with this metric come in when evaluating productivity or level of contribution to some project. If programmer "A" writes twice the number of lines as programmer 'B", he may or may not be contributing more - maybe "B" is working on a harder problem.

“My point today is that, if we wish to count lines of code, we should not regard them as "lines produced" but as "lines spent": the current conventional wisdom is so foolish as to book that count on the wrong side of the ledger.” - Edsger W. Dijkstra

Number of Commits

Commits are arbitrary changes captured in a single moment in time. Their size and frequency do not correlate with the work needed to achieve that change. At best, commits can be viewed as an indication of activity. It is an easy statistic to measure but it’s no more useful than recording the number of hours a developer worked during a week.

The general rule of thumb is that a programmer should be consistently making commits, so basically ‘a software engineer should write code…’. A Software engineer is paid to write code, so why should they be measured on how many small changes or how little big changes they make to that code.

Keystrokes

Ask yourself, would you measure a software engineers productivity on keystrokes? I certainly hope not as this is a lot worse than measuring the lines of code they write, again a software engineer is paid to write code, so that means they are paid to type. This would be the equivalent of saying that a bus driver is measured on how much far he drives the bus… in this case every variable name would not be short but more descriptive than the documentation for the code.

E.g. instead of:

int index = 0;   - Keystrokes = 14

It would be:

int the_index_of_the_first_for_loop = 1-1; - Keystrokes = 42

Test Coverage

The goal of a test is to find bugs. And if they didn't, they failed as tests. Instead of improving code quality, they might only be giving you a false sense of security. The code could still have bad commenting or inappropriate data structures. Code with absolutely no tests can be extremely high quality, readable, beautiful and efficient. From this it's not fair to say that code with 80 % test coverage is of higher quality than code with no test coverage. As a result tst coverage should not be considered as measurable data to base a software engineers performance.

Speed

Could you measure a software engineer based on their speed? In terms of finding problems and solving them, or even doing a complete U-turn on a project it's a good feature. But having your software team being measured by how quickly they work could lead to a quick burnout. Although speed would be a great feature it could also be a dangerous one. The amount of small mistakes or small bugs could escalate when speed is a top feature.

Platforms

- **Git**
- **The Active Badge Location System**

- Code Climate
- Code Beat

Git

For starters, there’s git. It’s a fantastic tool for development teams, and I’d imagine every development team would be using this as a toll. As we all know git tracks your commits and lines of code, as stated above these units are not good enough to measure a software engineer. It is more for branching and merging different sections of code to make it easier for the team.

The Active Badge Location System

Personally, after learning about the active badge location system, I would not be comfortable using it. Putting such a device into the workplace is viewed unfavorably on the grounds that it violates personal freedom and individual privacy rights. Although if other people have no problem with it then the system isn’t 100% accurate. As the system tracks the badges and not people, anyone could just leave their badge at their desk all day, rendering the badge useless as it gives false information on the employee's location.

The badge could also inherit more features such as a microphone. Although the microphone would be used for two-way communication, it’s like saying a webcam on a computer is only on when you want it to be, until you watch Snowden or Black Mirror then there’s a piece of tape over it. I can’t imagine any employee willing to wear a microphone around their neck so their boss could listen in.

GitPrime

First a review from a customer[2]

“Although we were only a small team of 2-3, we got a lot of value from GitPrime's stats and analysis. It really helped us to see where we were spending our time and assess the effects some of our decisions had on our code base. We found it to be great for management and for personal development and analysis of our work patterns.”

From the review it’s pretty obvious the customer was more than happy with Gitprime. It helps the customer know who’s doing their job or not by telling them who has/hasn’t contributed within a time limit, who contributed to what product and other visual data it can represent. It seems like a good tool to help teams work together and for a manager to overlook a project.

Case Study[1]

The following case study is focused on Camille Fournier, former CTO at Rent the Runway. She explains how she decided to measure the engineering team.

She decided to measure the engineers based on the current goals. Until you start to understand the actual cause of “velocity” measuring it only tells you that there might be some problems.

She asks what is it that makes a business succeed and for most companies the thing that brings success more than anything else is the frequency of releasing code. So start by measuring release frequency. Especially if it is scheduled at less than once a week or release dates are often missed due to issues preparing code. But if you spend all of the time working on internal tools and processes to release faster than you’ve missed the point. The point is not the perfect continuous deployment system. The point is enabling engineers to build features and release them easily so that your business can learn.

Another metric she mentions is the uptime of the system. If there are constant production outages or customers facing errors then it’s likely for this reason that customers are leaving. Plus it’s burning out the engineers from alerts and being on-call. Even if uptime isn’t reported, these metrics should be taken seriously as they involve the health of the team. Similar to releasing more frequently, improving uptime and reducing incident frequency often forces the team to fix things that are currently slowing down development of features.

Algorithmic Approaches

Pull Request

Pull Request is a code review service for businesses. It’s also simple to use as it is integrated with github. They run customized analysis for the specific team's needs and try to set defaults based on previous uses. But the main part of pull request is that they also get an expert reviewer and assign them to some code a business committed. They look through the static and linting outputs, generating a pull request for easy fixes, comments and questions for hard ones.

Although having an expert on demand to review items your normal developer wouldn't (security) sounds pretty cool. But wouldn't this lead to exposure of vital insider info of the company and its clients to outsiders? How will you make sure that it’s not misused?

A quote from the founder of PullRequest

“We’re using a 3-way innovation assignment/NDA that protects your code as if the

reviewer was a direct contractor with your organization. We do background checks on all reviewers, and we're limiting reviewers to markets where we have a presence for now."

Also

"We don't have our reviewers check the code out (it lives within a review environment). We also sign an NDA and PIA with all reviewers to ensure privacy."

So I guess this pretty much answers my question on how safe your code is the only question left is would you personally be comfortable with this.

Another issue is that sending your code off to some reviewer could be very time consuming, whereas you could just get the team to review each other's code. This would be more beneficial for the team as they would learn new technology, new ways of approaching a problem, they would understand part of the code they don't work with on a daily basis, they would have a bigger feeling of ownership etc. Except having code reviewed in a technology or a language no one in a company is an expert in could help developers have valuable feedback and ramp up faster.

Ethics

In terms of legal issues the only real thing an employer can't monitor is keystrokes. As keystrokes can contain passwords or other personal information and that would be a clear violation of privacy.

From an ethical point of view, I wouldn’t think employees mind employers monitoring whether they play computer games at workplace or not. Facebook use or non-work related sites browsing during paid hours monitoring is also an intrusion most of the staff would most likely agree to be reasonable.

Personally as long as the company states any monitoring they want to carry out on me in a legal agreement when I start working there then I’d have no problem with being monitored.

Although myself people would not be comfortable being monitored without a policy and therefore feel it is unethical. This can cause some problems in the workplace such as the atmosphere of mistrust and with strict policies can also block off any creative thinking. Which could result in the loss of employees

For both parties sake it is recommended that the employer keep these policies legal and ethical. This can create a better working atmosphere where everyone is happy.

References

[1] https://www.oreilly.com/ideas/ask-the-cto-measuring-team-performance

[2] https://www.capterra.com/p/153077/GitPrime/