Senan d'Art - 17329580

Measuring Engineering

Measurable Factors

There is a huge amount of different data that can be gathered on an individual software engineer. For this report I have broken it down into 3 categories: Code - data on the type of work the engineer is doing, Environmental - data on the work environment, and Personal - data specific to the individual being measured.

Code

The data gathered here could be gathered through a version control system such as Git.

Code Quantity

This is perhaps the most simple way to measure the amount of work done by an individual software engineer: a count of the number of lines of code written.

The concept of source lines of code or SLOC[1] involves measuring the number of lines of code written by an engineer. This metric however has several issues: simply measuring performance by the volume of code produced does not take into account the format of the code written, for example the following snippets are identical in their effect:

```
for (i = 0, i < some_num, i++)
        do_a_thing()
```

```
i = 0
while (i < some_ num){
        do_a_thing()
        i++
}
```

Simply measuring the volume of code produced by an engineer could lead to individuals padding their performance statistics by writing very verbose code.

Another way of measuring the volume of code produced is by measuring the logical lines of code or LLOC written[2]. This is a measure of the number of statements within the code, disregarding formatting. In the case of the previous example this could break down into:

```
for (i = 0, i < some_num, i++) # 4 LLOCs
        do_a_thing()           # 1 LLOC



# Total LLOC : 5
```

```
i = 0                      # 1 LLOC
while (i < some_ num){     # 2 LLOC
        do_a_thing()       # 1 LLOC
        i++                # 1 LLOC
}
# Total LLOC : 5
```

As this demonstrates, LLOC can overcome some of the issues with SLOC. However this metric is less transparent than simply measuring lines of code and it can still be cheated in various ways. It is also far more difficult to implement a fair way to count logical lines of code than physical lines.

1 https://en.wikipedia.org/wiki/Source_lines_of_code
2 http://www.projectcodemeter.com/cost_estimation/help/GL_lloc.htm

Bugs per Line[3]

One method used for analysing the quality of code is the measurement of how many bugs or errors exist in an average line of code. This is usually measured in bugs per 1000 lines of code and provides some insight into the quality of the software being written as well as the skill of the engineer writing it and their familiarity with the tools they are using.

Consistency of Code

This is a measurement of an engineer's performance over time.

One way to measure the consistency of work is to consider the quantity of code produced with respect to time. Ie evaluate SLOC or LLOC as a function of time. This information could be very useful in gauging a software engineer's performance but also suffers from several pitfalls eg. it does not take into account the difficulty of the work being done.

Another metric that can be used is 'churn rate' [4], this is the percentage of a developer's code that is an edit to their recent work[5] ie. how much of their code they are having to rewrite. A high churn rate can indicate low quality code being written whereas a low churn rate can indicate that the code written was of a higher quality. Churn rate can also be an indicator of how clear a project goal is as constantly changing objectives will lead to a higher churn rate.

Commit frequency could also be used to gain insight into the software development process. The number of commits in a day as well as the pattern of commits over time could be used to analyse performance. For example does an engineer commit very few times a day or does the amount of work they do reduce towards the end of the week.

Code Review

Pull requests or code reviews can also provide useful insight into work being done by a software engineering team. Since pull requests allow other members of a team to comment of code written by others it can be an opportunity to gain insight into systemic issues within the team[4] and also to find ways in which to improve the efficiency of the team.

During code reviews the readability of the code as well as how clearly it is commented can also be measured. While this may be quite subjective it could help improve processes by guiding the change of coding standards.

The way in which feedback is delivered to the original developer during a pull request can also be indicative of how well a team works together. An example would be determining whether the feedback is constructive or just critical.

Bug Fixes

Fixing bugs can often be a more involved process than writing the original code. The process requires an understanding of the problem code and how it interacts with other parts

3 https://www.mayerdan.com/ruby/2012/11/11/bugs-per-line-of-code-ratio
4 https://scottbarstow.com/churn-in-software-development/
5 https://stackify.com/measuring-software-development-productivity/

of the software. Some bugs can be very challenging to understand, let alone fix and this can lead to them being very slow to solve. During this time the quantity of code a developer produces could be quite low which makes their performance harder to measure as it is dependent on the difficulty of the problem which is again, hard to measure.

Environmental

The working environment can also be measured to assess its impact on software engineer performance.

Office Climate

Variables such office temperature and light levels or sources can have an impact on an individual's performance. Office temperature has a measurable impact on performance with temperatures too high or too low having a detrimental impact on productivity of people working in the environment[6]. The type and quality of lighting in the office can also can also have a significant effect on individual performance[7]. If the end goal of measuring performance is to improve it then these should also be considered.

Communication With Coworkers

Conversations between software engineers in the workplace could be analysed to gain insights from conversation tone or length[8]. Furthermore conversation content could be analysed to infer how focussed developers are on their work. This information also has the potential to provide some insight into team dynamics by providing detailed information on how colleagues treat each other in the workplace.

Personal

Health

Keeping track of various data relating to a software engineer's health could allow for a more detailed analysis of their performance and also improve health and productivity by providing the ability to both prevent and diagnose health problems that are currently affecting performance or could do so in the future.

Measuring posture while working could provide information on how focussed an engineer is and also provide recommendations to improve comfort, performance and long term health[9]. By combining this information with other data such as heart rate, breathing rate and blood oxygen levels, stress could be measured[10] and links established between stress and performance.

6 https://indoor.lbl.gov/sites/all/files/lbnl-60946.pdf
7 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4031400/
8 https://en.wikipedia.org/wiki/Humanyze
9 https://ijooes.fe.up.pt/article/view/2184-0954_002.002_0005/89
10 https://assets.firstbeat.com/firstbeat/uploads/2015/10/How-to-Analyze-Stress-from-Heart-Rate-Variability.pdf

Hobbies

What an engineer is doing in their free time could affect their performance at work. By gathering this information, a clear relationship could be established between engineering performance and activities outside of work. For example exercise[11] has a proven link to job performance. This could allow for improving engineer health and performance by providing insights into physical and mental health[12].

Computational Platforms

Cloud services

Frameworks

Ethical Concerns

Concerns about gathering data

Concerns about measuring data

Concerns about the interpretation of measured data

Computational Platforms

'Cloud' services
- Gitprime
- Waydev
- Velocity 2.0 by code climate

Frameworks
- Jasper
- Hackystat

Algorithmic analysis

AI / ML analysis
- Types of AI / ML

Statistical analysis

Ethics

Data gathered
- Personal data eg health data

[11] https://www.omicsonline.org/open-access/the-relationship-between-physical-exercise-and-job-performance-themediating-effects-of-subjective-health-and-good-mood-2223-5833-1000269.pdf

[12] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1470658/

- Communication

Analysis

- Is using a machine to measure a person okay?
- ML issues - use image recognition fudging example
- Does this data provide and accurate image of the situation?

Interpretation

- Does the person looking at the data understand the context?