Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU33012 SOFTWARE ENGINEERING.

Measuring software engineering report.

Terlo Akintola.

Contents.

1 Introduction.

Software engineering is a discipline that applies systematic applications of engineering approaches to the development of software.[1]
As such a software engineer is then somebody who applies the principles of software engineering to design, test and evaluate software.[2]
This report will delve into and explore methods of providing a basis for comparing and measuring the productivity of software engineering.

2 How can one measure engineering activity?

Software has a lot of moving parts and can be complex to analyse.
By definition a software engineering activity can be defined as measurable if
It can be described in specific terms usually expressed as a quantity.[3]
Software engineering can be measured using multiple variables as outlined in 2017 research by Edson Oliveira et al 2017.
The most popular ways of measuring software engineering from the context of the project is by lines of code to “effort” where effort is measured as the rate of output per unit of input.To measure the software engineering activity for a developer the most popular method was the ratio of lines of code to time.[4]

2.1 lines of code

Lines of code are a measurable quantity to calculate the amount of work a software engineer produces and In some instances can be a way to measure who is the most productive engineer (at least in terms of producing lines of code).
On the surface this can be a simple solution to find the most productive engineer, but for many reasons I believe this metric has many fatal flaws when used as a single metric or in a ratio to other metrics, such as time.

1 Abuse.
If productivity is measured in terms of how many lines an engineer can produce and they are encouraged to produce more lines, then it is almost inevitable abuse of the metric will soon follow and as such tasks that should at most be X amount of lines

become X+Y amount of lines.As such both of these examples bellow produce the same output

```
1.  print(hello)
 >>hello
```

```
1. print(h)
2. print(e)
3. print(l)
4. print(l)
5. print(o)
 >>hello
```

2. language differences.

If multiple engineers are working with different languages on an application, it puts each engineer on an unlevel playing field. Those who contribute on HTML will easily contribute more lines of code with less effort than those who contribute Java or python.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Hello, world!</title>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <meta name="description" content="" />
</head>
<body>
  <h1>Hello World!</h1>
</body>
</html>
```

java

```
1.  Public class Test{
2.          Public static void main(String[] args){
3.                  System.out.println(“Hello World”)
4.          }
5.  }
```

python

```
    2.  print(“Hello World”)
```

2.2 Number of commits

Number of commits (on its own) to a subversion system, similar to measuring by lines of code this is an abusable metric and there is no doubt that these metrics can actually leading to inefficient work as engineers **can** spend time creating commits that have no true value to the project for the sole purpose of increasing “productivity”. A much better metric would be time taken to complete a commit where the number of required commits are assigned to different tasks prior. This method incorporates an element of **lead time** where this is defined as the time between when the task was assigned to when it was completed. Although this is not absolute as some tasks by their nature require more time to complete and as such wouldn’t necessarily make one engineer “better” than another when comparing commit/unit time.

3 Platforms for data collection and analysis.

3.1 utility cloud computing

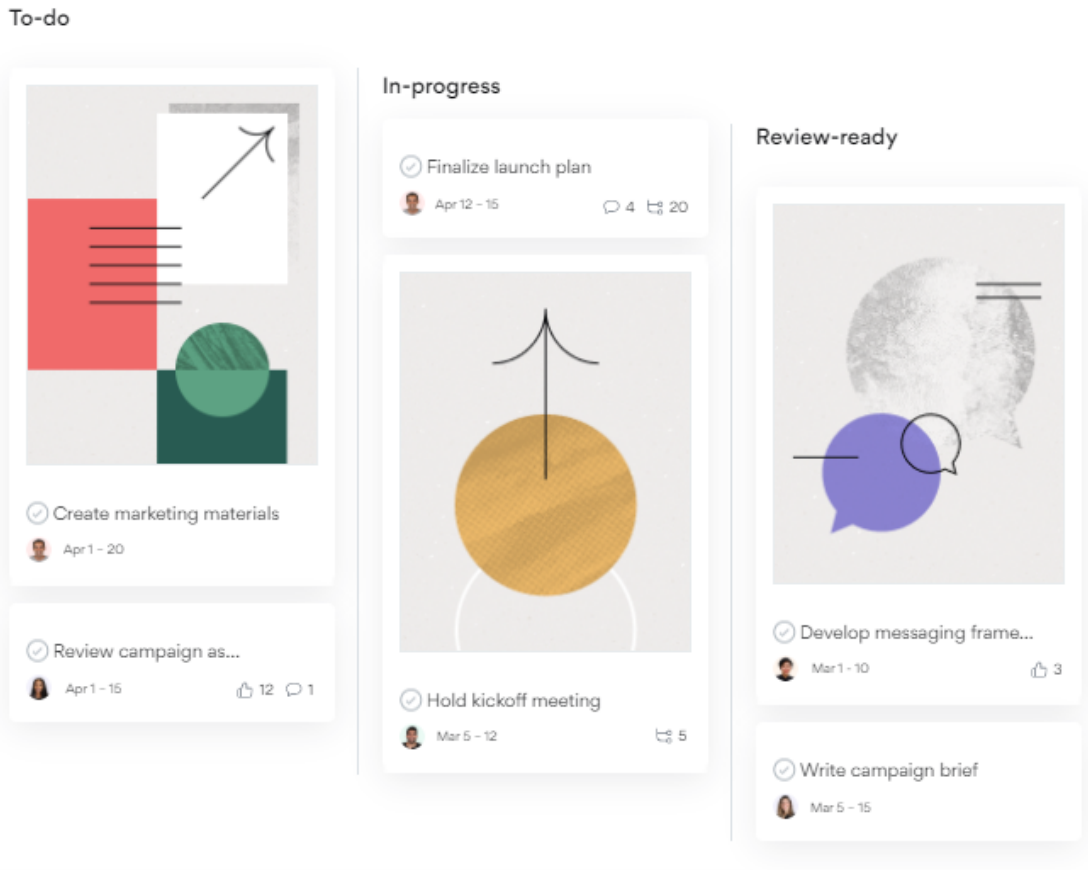
Utility computing offers a service of storage and power resources and the customer is charged based on how much they use the service. Depending on the type of services offered this may be called known as:

- I.a.a.s - infrastructure as a service
- S.a.s.s - software as a service
- H.a.s.s - Hardware as a service.[5]

There are many Utility computing platforms which offer data collection in databases and analytics which can aid in measuring software engineering by the ability to compute complex calculations. The most popular examples of utility cloud computing providers are AWS, Microsoft azure, Google cloud, Alibaba cloud and IBM cloud.[6]

3.2 Employee Collaboration tools

Employee collaboration tools such as Asana, Trello, Monday provide companies with the ability to keep track of progression in the project at hand. As such these tools provide a metric of output/time companies are able to find inefficiencies and improve and make informed decisions based on the data that they have gathered.



4 Methods of computation

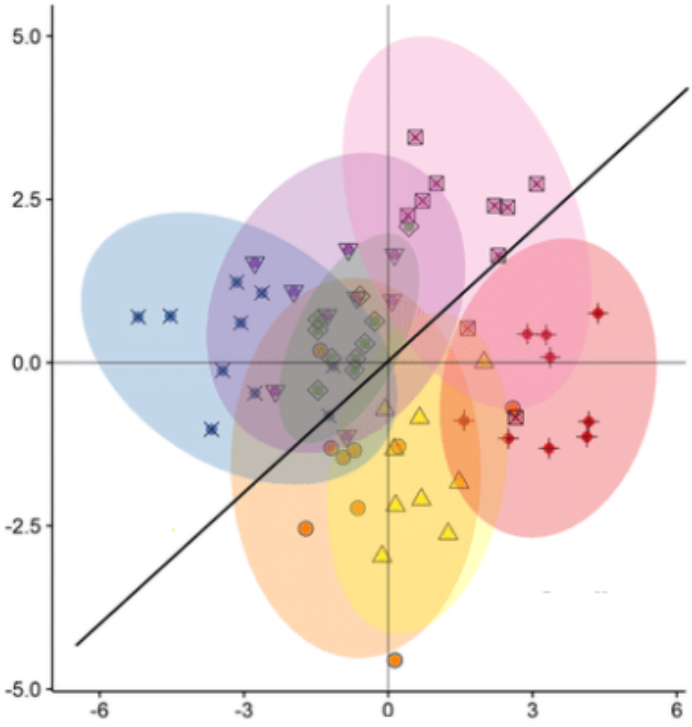
4.1 simple counting

The most basic form of computation is counting, by simply counting occurrences of metrics X and Y that pertain to a software engineer you can build a profile about the engineer. This can be the number of commits to a repo, number of lines covered in test code or hours spent working on the project at hand.

The pros of this method of computation is that it is simple to execute.but the cons are that it is time consuming and inefficient, such that a computer would be able to carry out larger and more complex computations at a faster speed.

4.2 Multivariable analysis

Multivariate analysis would be an effective method for computation for which the performance of a software engineer can be measured.Pattern recognition methods like Linear Discriminant analysis would allow us to profile software engineers based on their performance over a multitude of variables such as Coverage of code(%) per test written, Time taken (in days) to bring a task to completion.Number of commits to github/day (although this is an abusable metric) , ratio of additions to deletion of lines of code per commit.



[7]

4.3 Linear Regression.

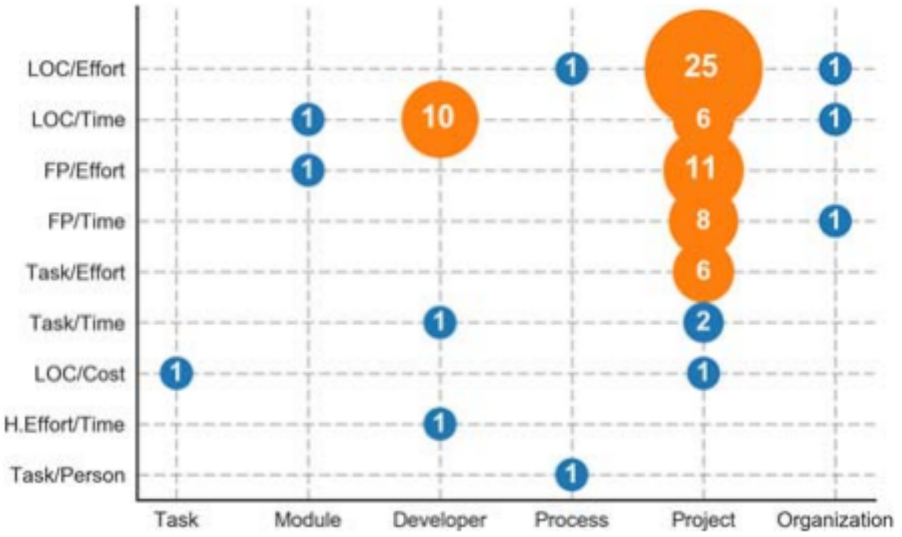
If we have multiple metrics about a software engineer and between any two of the metrics X and Y there exists:

- Linearity, A linear relationship between the independent variable and the dependent variable.
- Independence, independent between the residuals.
- Homoscedasticity.

- Normality

Applying a linear regression model would be a good predictor of what would be expected of a software engineer in a given metric X when provided another metric Y once the relationship between X and Y follows the above expectations[8].

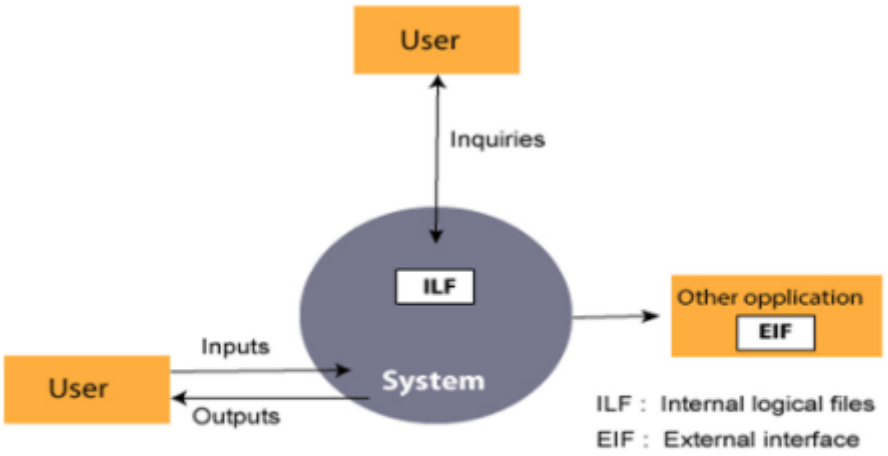
4.4 Function point analysis



[4]

According to the research by Edson Oliveira et al 2017, Function points relative to effort and relative to time were the very popular methods of measuring software engineering in the context of the project [4]

Functional point computation was developed by Allan J albrecht in 1979 at IBM.
It is an effective method of measuring software engineering as It is used to calculate the cost of development early in the process.By using the parameters, Number of External inputs, External Outputs External inquiries,number of internal files and the number of external interfaces to algorithmically calculate a Function point value.[9]



FPAs Functional Units System

[9]

The benefits of using function point computation is, it is specification based and language independent.[9]

5 Ethical concerns

As with any processing of data about ourselves there exists many ethical concerns, these concerns arise as it's likely the processing of said data can affect us directly or indirectly.There are times in which processing of our data using AI models can benefit us such as using our data provided to services that help us find out about information which is relevant to us based on how we are profiled. Simultaneously there are also clearly times in which processing of our data does not serve us or others in a way that is beneficial to either the data subject or data handler for example Amazon's recruitment AI which was biased against women because of the bias of the majority of the training data being men.[10] The data gathered and processed to measure software engineering is no exception and can be flawed as a result of bias or data inconsistencies irrespective of if the data is about a software engineer or software project.While regulations in place can provide a level of

protection for us as data subjects they do not offer any guarantee that the data being processed will not be misused or even compromised.

6 Conclusions

In conclusion, there are many methods to quantify the inputs and output of a software project/engineer and come to conclusions based on this data but I don't believe, from an objective standpoint that we will be able to "objectively" measure software engineering in the context of software engineers/projects as there are in theory an infinite number qualities of a software engineer which are not quantifiable or difficult to quantify but I do believe that we should still seek to find more innovative methods to measure software engineering as these can allow us to build a good picture for what can be considered "good" software engineering activity.

7 Sources

[1]https://en.wikipedia.org/wiki/Software_engineering

[2]https://en.wikipedia.org/wiki/Software_engineering

[3]https://www.merriam-webster.com/dictionary/measurable

[4]https://www.scitepress.org/papers/2017/63144/63144.pdf

[5]https://vmblog.com/archive/2019/09/25/what-is-utility-computing-in-cloud-computing.aspx

[6]https://asana.com/

[7]https://www.researchgate.net/figure/Linear-discriminant-analysis-LDA-A-score-plot-and-B-biplot-showing-overall-fatty_fig4_322766513

[8]https://www.statology.org/linear-regression-assumptions/

[9]https://www.javatpoint.com/software-engineering-functional-point-fp-analysis

[10]https://www.independent.co.uk/life-style/gadgets-and-tech/amazon-ai-sexist-recruitment-tool-algorithm-a8579161.html