Measuring Software Engineering

Masanari Doi

Student ID 19313167

Contents

1. Introduction

2. How can software engineering be measured?

In this section, this essay will discuss measuring the productivity of software engineering. Regarding the benefits of measuring productivity, according to Steve McConnell(2021), “organisations would need to measure how productive their Software Engineers teams are, in order

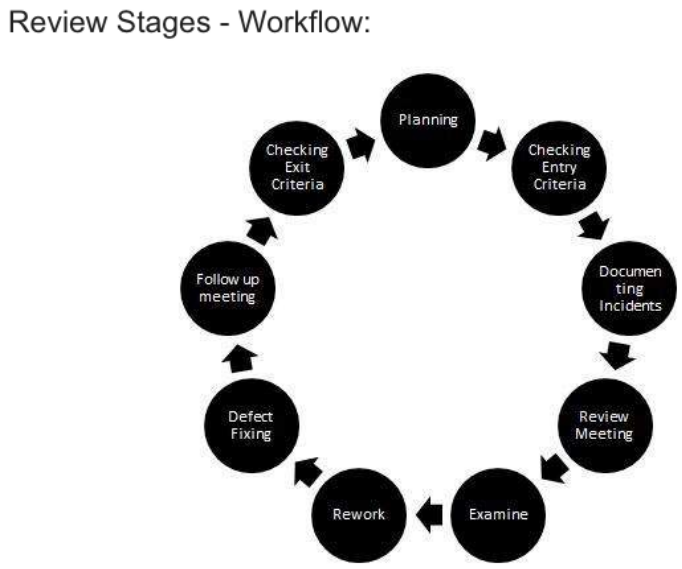
- To develop competitive analyses and benchmarks
- To track progress over time
- To reward high performers
- To determine resource allocation
- To identify and spread more productive development processes across the organization”

It can be said that measurement of software engineering is essential to improve the efficiency of the work in the organisation.

2.1 Reviews

One of the examples of software engineering measurement methods is reviewing the work. This is defined as a systematic examination of a code or a work by the people in the team for decreasing errors or bugs in an early stage of the software development life cycle. Documents such as code, requirements, system designs, test plans and test cases are verified by this review method.(tutorialsPoint, no date) By utilising this method, it has been said that efficiency and productivity of a development team is enhanced and it becomes possible to detect defects in initial stages, which improves timescales. Moreover, it can be said that, because of enough time conducted in the early stages, testing costs and time are decreased.(tutorialsPoint, no date) Therefore, if the

developers appropriately conduct reviewing, find some errors and improve the work, it would be ture that the team work would be more efficient and productive.



(figure 1, review Stages, tutorialsPoint, no date)

2.2 Lines of code

Employing lines of codes, it would be possible to examine the size of the project, and the lines of the codes is easy to check. Moreover, the code having shorter lines would be thought easy to interpret. (Peipman, 2010) However, for some aspects, it is said that the lines of code is not a useful method to measure the productivity of the code. For example, some code needed to be read by a teammate who needs to understand how the code is working or where the developmental point is. However, lines of code can not show us these points. Therefore, it would be true that to concentrate on the aspect of easiness of the code is more important than the lines of codes. (Opidi, 2020) Take two javaScript codes which have else-if statement below for example.

```
const firstNumb = 100;

let secondNumb;

if (firstNumb > 50) {
  secondNumb = "Number is greater than 50";
} else {
  secondNumb = "Number is less than 50";
}
```

```
const firstNumb = 100;

let secondNumb;

const secondNumb = firstNumb > 50 ? "Number is greater than 50" : "Number is less than 50";
```

(Figure 2 & 3 , sample JavaScript Code, Opidi, 2020)

These two codes do the same work but the first one clearly has many more code lines. However, it can be said that the first one is easier to translate than the fewer codes of the second one. It would be needed to make the code readable because making unreadable codes seems to let team members make much effort to understand the code and decrease the efficiency. Therefore, it would be true that the lines of code cannot provide the detailed information, such as how much easy or comlexed the code is. Moreover, it can be more practical and important to fucus on readability rather than the number of lines of codes. (Opidi, 2020)

Having said that, as an advantage of shorter lines of code, it can be true that if we only need one line for the if-else statement, the code in figure 3 can save time. Moreover, it can be obvious that fewer lines of the codes is more efficient than more lines because it would increase the possibility of causing bugs to delete, and finding them seems much more difficult in a long code. It may be true that the fewer lines decline the amount of tasks such as decreasing the bug count. Furthermore, the long lines make a huge amount of variables that the developer have to name, which would take long time and

make other developers confused. Therefore, if the code is readable and easy to interpret, shorter lines of code seems preferable. (Opidi, 2020) However, if it can be possible to know the size of the project (even if it has shorter lines), it does not tell us how much it is easy or complicated to understand. As it is mentioned, only if the code is easy to read, the lines of code should be taken into account by developers to increase efficiency. It is impossible to determine whether the codes are interpretable only by seeing the number of lines of the code. Hence, the lines of code would not be regarded as a useful method.

3. Platforms

3.1 Microsoft Azure

3.2

what is possible here, and also what specialist infrastructure has emerged to perform various kinds of data analysis.

4. Computation

5.Ethics

6. Conclusion

7. Bibliography

1. Microsoft.(no date). *Data Lake Analytics*. [online] Available at: https://azure.microsoft.com/en-in/services/data-lake-analytics/#overview [Accessed 27 Dec. 2021].

2. Opidi, A.(2020). *Long Code vs. Short Code: What's Better for Your Use Case?*. [online] Available at: https://www.freecodecamp.org/news/long-code-vs-short-code/ [Accessed 20 Dec. 2021].

3. Peipman, G.(2010). *Code Metrics: Line of Code*. [online] Available at: https://gunnarpeipman.com/code-metrics-lines-of-code-loc/ [Accessed 24 Dec. 2021].

4. Tutorials Point.(no date). *Code Metrics: Line of Code*. [online] Available at: https://www.tutorialspoint.com/software_testing_dictionary/review.htm [Accessed 2. Jan. 2022].

5.