

CSU33012 Software Engineering

Measuring Engineering Report

**Stephen Davis, Std# 18324401**

**November 14, 2021**

Contents

[1 Introduction 2](#_Toc88749466)

[2 How Can Software Engineering be Measured? 2](#_Toc88749467)

[2.1 Lines of Code 2](#_Toc88749468)

[2.2 Number/Frequency of Commits 2](#_Toc88749469)

[2.3 Number of Code Reviews Done 3](#_Toc88749470)

[2.4 Leadtime 3](#_Toc88749471)

[3 What Platforms Can Be Used to Gather and Process Data? 3](#_Toc88749472)

[3.1 Pluralisation (Git Prime) 4](#_Toc88749473)

[3.2 WayDev 4](#_Toc88749474)

[3.3 Code Climate 4](#_Toc88749475)

[3.4 Hackystat 4](#_Toc88749476)

[3.5 Personal Software Process 4](#_Toc88749477)

[4 What Algorithms Can We Use? 4](#_Toc88749478)

[4.1 Halstead Complexity Measures 4](#_Toc88749479)

[4.2 Artificial Intelligence 4](#_Toc88749480)

[4.3 Computational Intelligence 4](#_Toc88749481)

[4.4 Cyclomatic Complexity 4](#_Toc88749482)

[5 Is This Ethical? 5](#_Toc88749483)

[5.1 Privacy 5](#_Toc88749484)

[5.2 Increasing Productivity 5](#_Toc88749485)

[6 Conclusion 5](#_Toc88749486)

# Introduction

This report will discuss why one would want to measure software engineering, how software engineering can be measured, what platforms can be used to measure software engineering and what algorithms we can use to carry out such measuring. This report will then discuss the ethics of measuring software engineering, before finally concluding.

# How Can Software Engineering be Measured?

There is much debate about whether software engineering can actually be measured. A popular view, and my personal view, is not whether software engineering can be measured, but how can software engineering be measured? It is worth noting that there are many ways we can measure software engineering, but not every way is accurate, insightful and fair. Most of the time the issue with these inaccurate measurements is that they lack context. It is worth noting that each measurement comes with its own advantages and disadvantages. Some examples of how to measure software engineering productivity which I will now discuss are:

* Lines of Code
* Number/Frequency of Commits
* The Number of Features Delivered
* Number of Code Reviews Done
* Leadtime – how long it takes you to go from idea to delivered software
* Total hours of downtime – dice used to count time spent on activity

## Lines of Code

The most obvious method to measure software engineering is to measure the number of lines of code a software developer writes. This metric is considered to be “The Original Code Metric” (<https://www.gitclear.com/blog/the_4_worst_software_metrics_agitating_developers_in_2019> ). The advantages for this method are that it is very simple to measure and easy to understand this measurement. You do not need a technical background to understand how to analyse this metric.

Although there are some advantages to using Lines of Code (LOC) as a measurement, they are far out-weighed by the disadvantages. The most important disadvantage of LOC, is that it is “a very easy metric to game” (<https://www.pluralsight.com/blog/teams/lines-of-code-is-a-worthless-metric--except-when-it-isn-t-> ). Figure 1 and 2 below illustrates how one could gamify their code to appear more productive.

Figure (for loop)

Figure (same as above by copying and paste same line of code)

As can be seen by the above examples, LOC as a metric promotes inefficient, sloppy code and is far too easy to game. We must remember, “You don't pay a Michelangelo to make brush strokes, you pay him to be a genius” (<https://www.pluralsight.com/blog/teams/lines-of-code-is-a-worthless-metric--except-when-it-isn-t-> ).

## Number/Frequency of Commits

At first, measuring the number of commits made by an engineer may appear to be beneficial as a metric for measuring productivity, however, just like LOC, it is easily gamified. Our hope would be that software engineers commit code when they have made progress on a project, and by comparing the number of commits made by different developers, the higher the number, the more productive an engineer is.

However, in theory, two engineers could essentially write the same code, except the first engineer commits the code when he has the functionality working, and let’s say the second engineer commits every time he adds a new line of code. Now, of course, we do not want the latter to happen, but what is stopping this engineer, or any engineer, from doing so? Again, this illustrates how gamification can easily be utilised to improve your score when being analysed by the number of commits you make.

The disadvantages of using the number of commits made by an engineer is that it encourages insignificant and unnecessary commits. “You are incentivising them to make a commit every time they author a line of code” (<https://www.gitclear.com/blog/the_4_worst_software_metrics_agitating_developers_in_2019> ). The size and value of the commit is not taken into account, and this is why it is so easy to gamify. Perhaps one of the cruellest disadvantages of using the number/frequency of commits as a metric is that “if you're a hard-working developer that's striving to solve as many issues as possible, by simply saving their work more often, your lazier coworker will shoot past you on the commit count leaderboard?” (<https://www.gitclear.com/blog/the_4_worst_software_metrics_agitating_developers_in_2019> ). This metric rewards those who can game the system and those who play fairly appear to be the workers lacking in productivity. Simply put, “Commit count creates a toxic atmosphere among developers”.

One advantage of using this metric is that if you notice it has been a while since an engineer has made a commit, this is “often a signal that they may be stuck” (<https://www.gitclear.com/blog/the_4_worst_software_metrics_agitating_developers_in_2019> ). However, it is quite clear that this one advantage is negligible when compared against all the disadvantages associated with this metric.

Again, as software engineers, we favour quality of code over quantity.

## Number of Code Reviews Done

## Leadtime

“Lead time quantifies how long it takes for ideas to be developed and delivered as software” (<https://stackify.com/track-software-metrics/> ). To measure Leadtime, you “need to have a clear definition of when work begins and ends” (<https://humanitec.com/blog/lead-time-a-key-metric-in-devops> ). The idea behind using Leadtime as a metric is to measure how quickly engineers or a team of engineers can go from idea to production. If we notice one project idea has a significantly longer Leadtime than another, we may ask why? Perhaps a project idea was underspecified or there were permission requests which took a long time to come back. The advantages of this metric are that it incentivises software engineers to “improve how responsive they are to customers” (<https://stackify.com/track-software-metrics/> ). Another advantage is that a company may realise from analysing this metric, that their Leadtime for releasing a fully-functional working version of the idea is too long, and their competitors may have beat them to the punch. By analysing the Leadtime, team leads may decide to “release little and often; you release each feature as it is ready, rather than waiting for an “all-in” big release” (<https://humanitec.com/blog/lead-time-a-key-metric-in-devops> ). This adapted approach would then keep the customers engaged and loyal to your company’s product.

With any measurement, if you measure the wrong things, you risk pushing the quality of code down, which tends to reward mediocrity. Therefore, it is vital that if you are measuring software engineering, you choose a fair, insightful and well-thought-out metric. Do not just measure software engineering for the sake of measuring.

# What Platforms Can Be Used to Gather and Process Data?

There are various platforms which can be used to gather and process data in our search to measure software engineering. The examples I will discuss in this report are:

* Pluralisation (Git Prime)
* WayDev
* Code Climate
* Hackystat
* Personal Software Process

## Pluralisation (Git Prime)

## WayDev

## Code Climate

## Hackystat

## Personal Software Process

# What Algorithms Can We Use?

* Halstead Complexity Measures
* Artificial Intelligence
* Computational Intelligence
* Cyclomatic Complexity

## Halstead Complexity Measures

## Artificial Intelligence

## Computational Intelligence

## Cyclomatic Complexity

# Is This Ethical?

* Moral Issues - Privacy
* Increasing Productivity

## Privacy

## Increasing Productivity

# Conclusion

Source:

<https://blog.pragmaticengineer.com/can-you-measure-developer-productivity/>

<https://www.gitclear.com/blog/the_4_worst_software_metrics_agitating_developers_in_2019>

<https://www.pluralsight.com/blog/teams/lines-of-code-is-a-worthless-metric--except-when-it-isn-t->

<https://stackify.com/track-software-metrics/>

<https://humanitec.com/blog/lead-time-a-key-metric-in-devops>