# Software Engineering

# Measuring software Engineering report


## Iraklis Bogiatziou

## Student ID: 18329647

In an era where every choice counts, employers and employees are always looking for methods to make work more efficient. The former want to track the productivity of their engineers in order to identify and resolve weaknesses so that they maximize profit in the long run. Employees, who want to keep employers and clients satisfied, want to keep track of their performance so that they are capable of meeting deadlines and be on par with the employer's expectations.

However, to put it simply, measuring productivity is not an easy task. Apart from the lack of a definitive metric to measure development, projects also include factors such as unit testing and code refactoring which are essential during development but are hard to measure. With the development of new technologies and with software engineering becoming more and more popular, different approaches have arisen, each offering their own solution to the problem.

This report is divided into four sections. To begin with, I will outline some of the difficulties in measuring software engineering. Then I will explore different platforms that provide measuring techniques as well as algorithmic approaches that can be used in the industry. In the end, I will discuss the ethical concerns that may arise from those approaches.

## Metrics

### Commits

Developers use commits to apply changes to a source code in a repository. For this reason, it can be used as a productivity measurement technique as employers can view the number of commits of each developer. At first glance, it seems like the more commits a programmer has, the more "productive" they are.

However, we can easily understand that this method can be misinterpreted as each developer, who is part of a team, differs and does not follow the same commit practices as others. “The frequency of the commits, nor the size should be compared between programmers. What should be committed is often a matter of personal preference and is subject to debate.” (Job van der Voort, GitLab). The size and frequency of commits do not correlate with the completion of a task. Relying on this could lead developers to focus on unnecessary commits rather than software development which could decrease work performance and efficiency.

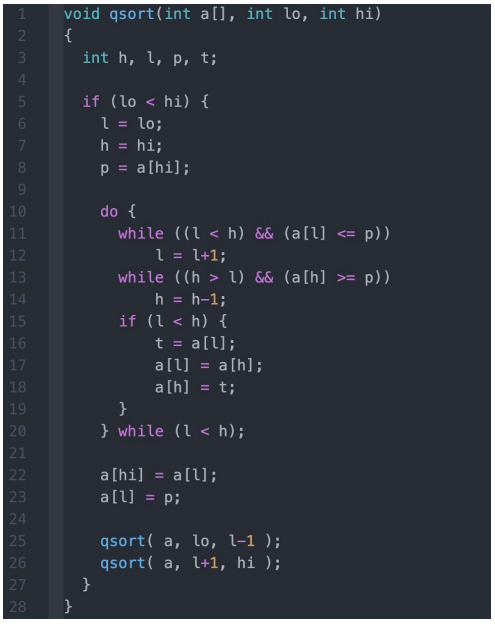
## Lines of Code

Another method of measurement could be the Lines of code (LOC) where the number of lines a developer has produced correspond with the productivity. Although it is a metric that has been used for many years, it certainly comes with many flaws. As a problem can be solved in many ways, developers could favor a longer solution instead of a short one. Such an approach could lead to duplication and inefficiency, as shorter code is usually more optimized and factored in. Secondly, LOC itself only looks for the number of lines and does not take into account other factors, such as the language. Different programming languages can implement methods in far fewer lines of code than others. For example, what can be written in Haskell using one line of code may consist of multiple lines in C, and thus, LOC could wrongly favor the C developer.

Quicksort as per the Haskell website:

```
quicksort [] = []
quicksort (p:xs) = (quicksort lesser) ++ [p] ++ (quicksort greater)
    where
        lesser = filter (< p) xs
        greater = filter (>= p) xs
```

Quicksort in C:



```c
void qsort(int a[], int lo, int hi)
{
  int h, l, p, t;

  if (lo < hi) {
    l = lo;
    h = hi;
    p = a[hi];

    do {
      while ((l < h) && (a[l] <= p))
          l = l+1;
      while ((h > l) && (a[h] >= p))
          h = h-1;
      if (l < h) {
          t = a[l];
          a[l] = a[h];
          a[h] = t;
      }
    } while (l < h);

    a[hi] = a[l];
    a[l] = p;

    qsort( a, lo, l-1 );
    qsort( a, l+1, hi );
  }
}
```

## Ticket Driven Development

Another method of measuring software engineering is ticket driven development. With this approach, every ticket created represents a new task. When the task is completed the ticket is finally closed. For a section to be defined as complete, it should be bug-free as well as pass all the necessary tests. Ticket driven development could help developers concentrate on the outcome of the project rather than how many lines of code they produce or how active they look with commits. Despite this, this method is not flawless. With the developers in control, they could just create tickets consisting of smaller, less complex tasks, creating a sense of artificial productivity. Secondly, this does not necessarily give any information about the quality of the code. However, it is useful to see how quickly tickets come up and are closed during a project. Overall, if not misused, it can be a very effective tool for increasing productivity and speeding up development.

## Platforms

Being a project manager is not easy especially if all the code needs to be analyzed manually. This is why several platforms have been created to help teams keep track of their work. Tools ranging from software to actual physical devices are available to help optimize their coding practices. Some of them rely on human input whereas others are fully automated. A few of the most popular ones are:

- Pluralsight Flow
- Jira
- Waydev
- Timeular

### Pluralsight Flow

Pluralsight Flow, previously known as GitPrime, analyzes all the data that can be taken from Git into easy to understand reports for managers and engineers to increase their productivity. It combines some of the metrics mentioned previously which can be hard to understand on an individual level. Flow can show a developer's performance on a project by viewing their commits, the tickets linked to them as well as the pull requests that are currently out. Taking into account the language that is used on each project, it can more accurately portray each developer's impact. This is data that is easy to use from managers and team leaders. It can increase visibility about team contributions, indicate the biggest impacts, and give feedback in areas that are lacking behind. Flow isn't only useful for managers. It offers retrospective tools for developers to get a better understanding of where and why they were less productive.

### Jira

Jira is a project management software that, among others, leverages skills for software development, bug tracking, and agile software development. It is a Kanban system that offers boards of tasks separated into stages marking their current stage of completion. Jira also features Scrum boards for developers to keep track of their sprints and backlog to measure the velocity of the team. Roadmaps are also a great feature as they allow developers to plan, and have a clear vision of what they are creating. Jira offers many Agile metrics and does a good job reporting in capabilities in a detailed, but yet, easy to understand manner.

### Waydev

Waydev analyzes a developer's codebase from most major version control systems and generates reports and metrics to monitor, measure, and assess their efficiency, productivity, and performance. Instead of requiring developers to manually input data for sprint and workload reporting, as is the case with Jira, it creates automatic reports without the need for any input. When developers push code to a project, Waydev can identify whether the developer was making new work, fixing bugs, or helping others. This offers an advantage over Jira as developers are less likely to trick the system.

### Timeular

Timeular is an 8-sided dice which combines gamification with tracking time. The dice itself does not have access to any code or repositories but instead helps developers (and not only) keep track of their tasks. Each side can be assigned to an activity which can then provide valuable insights to improve

how to spend time more efficiently. This data can be assessed by project managers to view weak points and tasks that take the most amount of time.

Each platform and system comes with its advantages and disadvantages as they all rely on different metrics. Each company may have its view on how to measure efficiency. Some may prefer a more manual system where they would need the developers' input, whereas others would rely on automation.

## Algorithmic Approaches

The previous sections covered data accumulation using commits, Lines of Code, etc. Although these metrics offer simplicity and are straightforward, businesses are interested in more accurate and in-depth analysis of a project including its cost and schedule. I will outline some of these algorithmic approaches below.

### Analogy and Extension from Analogy

Analogous estimating is the technique of using former projects to compare them to current ones to determine the estimated cost and time of future projects. It measures aspects of different tasks such as complexity and size and since it is centered around comparisons, more data provides more precise results. Creating a database from this data can deem beneficial in many ways. Managers would be able to prepare for future projects taking into account previous costs and past experiences. Individuals and developer teams can estimate the effort and duration of their tasks too.

## Analogy-X

The Analogous estimation assumes that the new project is similar to the completed ones. Thus, it fails to provide accurate data when the new project is irrelevant to what already exists in the database. Analogy-X builds on this by using randomized tests and constructing matrices that are used to determine if the datasets are relevant. As a result, it eliminates irrelevant past projects and identifies the most appropriate ones.

## COCOMO Model

Cocomo is a software estimation model based on LOC which predicts various parameters using the size of the software. Unlike LOC by itself, it does not only rely on how many lines of code exist. It uses two key parameters to determine the quality of projects, Effort, and Schedule. The former corresponds to the amount of work for the completion of a task and is expressed in person-month and the latter means the amount of time needed to complete a task and is measured in units of time.

The model can appear in various modes, depending on the accuracy that is required. These modes are mentioned below.

**Organic:** Projects that are suitable for adequately small team sizes and simple problems.

**Embedded:** Projects that require a high level of complexity and large team sizes.

**Semi-Detached:** Projects in which the team-size, knowledge, and difficulty of the problems can vary between organic and embedded projects.

COCOMO also consists of three types of models, each with a different level of detail.

**Basic Model:** This level is used for quick, cheap, and fast calculations of the project cost. This, however, translates to a low level of accuracy

**Intermediate Model:** This COCOMO model takes into account other factors and data which take longer to implement and offers a more precise estimation than the previous model.

**Detailed Model:** As its name suggests, it is the most detailed out of the three models and offers the most accurate estimates.

Overall the COCOMO model offers an easy estimation of the total cost of the project by taking into account multiple factors from which the user can produce a very accurate result.

## Ethics concerns

Whenever data is collected, ethical concerns are always on the table about how that data is gathered, where it is stored, and how it is being used. The job of a monitoring system is to keep track of each employee and their performance in the company. For developers, it would mean using one of the platforms mentioned above, or any other system, for companies to determine if a project is going as planned or a developer is lacking behind and thus taking appropriate actions. Monitoring systems can be easily abused by employers, keeping track of their employees' every single movement. This however, is often overlooked when companies offer amenities such as gym memberships, meals, discounts, and other perks in return for data collection.

In my opinion, with GDPR in place, in Europe that is, abuse of data is not that easy. With that being said, I believe that data, if properly handled, can not only help developers perform more efficiently but can provide the foundations for research in the future that could alter software engineering as we know it today.

## Conclusion

To conclude, it can be argued that measuring software engineering is not an easy task as every approach is prone to errors, be it from human dishonesty/mistakes or the systems themselves. For the most accurate result, teams should balance this data with team meetings and other face to face methods. It is important to keep in mind that every developer and team is different and projects do not always follow the same pattern. These should be considered during every phase of development in order to maintain measurement accuracy.

## Bibliography


Circei, Alex. “What is Waydev?” *Waydev.co*, 2020, https://docs.waydev.co/en/articles/2878408-what-is-waydev. Accessed 18 November 2020.

Fowler, Martin. “CannotMeasureProductivity.” https://martinfowler.com/bliki/CannotMeasureProductivity.html. Accessed November 2020.

Haskell. “What's good about functional programming?” *Haskell.org*, 2020, https://wiki.haskell.org/Introduction#Quicksort_in_Haskell. Accessed 20 November 2020.

Koulla Moulla, Donatien, and Kolyang. *COCOMO model for software based on Open Source: Application to the adaptation of TRIADE to the university system*. vol. 5, ENGG Journals Publications, 2013. *Researchgate*, https://www.researchgate.net/publication/272819461_COCOMO_model_for_software_based_on_Open_Source_Application_to_the_adaptation_of_TRIADE_to_the_university_system. Accessed November 2020.

Malathi, S., and S. Dr. Sridhar. *An algorithmic approach for the implementation of analogy–X for software cost estimation*. Coimbatore Institute of Information Technology, 2004. *Researchgate*, https://www.researchgate.net/publication/288989234_An_algorithmic_approach_for_the_implementation_of_analogy-X_for_software_cost_estimation. Accessed November 2020.

Pluralsight. “INTRODUCING PLURALSIGHT FLOW: YOUR KEY TO DATA-DRIVEN ENGINEERING LEADERSHIP.” *Pluralsight.com*, 2019, https://www.pluralsight.com/blog/platform/pluralsight-flow. Accessed 18 November 2020.

Pluralsight. “What is Flow exactly?” *Pluralsight.com*, 2019, https://help.pluralsight.com/help/what-is-flow-exactly. Accessed 15 November 2020.

Van der Voort, Job. “Commits Do Not Equal Productivity.” https://about.gitlab.com/blog/2016/03/08/commits-do-not-equal-productivity/. Accessed November 2020.