# Software Engineering Report


Figure 1

## Abstract

The purpose of this report is to analyse how the software engineering process is measured. This analysis will be primarily focussed on four sub-headings:

1. Measurable Data
2. Computational Programs
3. Algorithmic Approaches
4. Ethics of this Measurement

This report is written using a mixture of academic papers and online resources on the topic of software engineering, as well as information discussed in the CS3012 module.

# 1. Introduction

## 1.1. Software Engineering

According to The Economic Times: *"Software engineering is a detailed study of engineering to the design, development and maintenance of software. Software engineering was introduced to address the issues of low-quality software projects."*

Simply put, software engineering is how software is designed and implemented to solve a given problem. There are now millions of software engineers worldwide, each looking to fully optimise solutions to problems. They attempt to do this through the software engineering process.

## 1.2. Software Engineering Process

The software engineering process was put in place to optimise certain aspects of the industry, namely:

- Analysis
- Design
- Coding
- Testing
- Maintenance

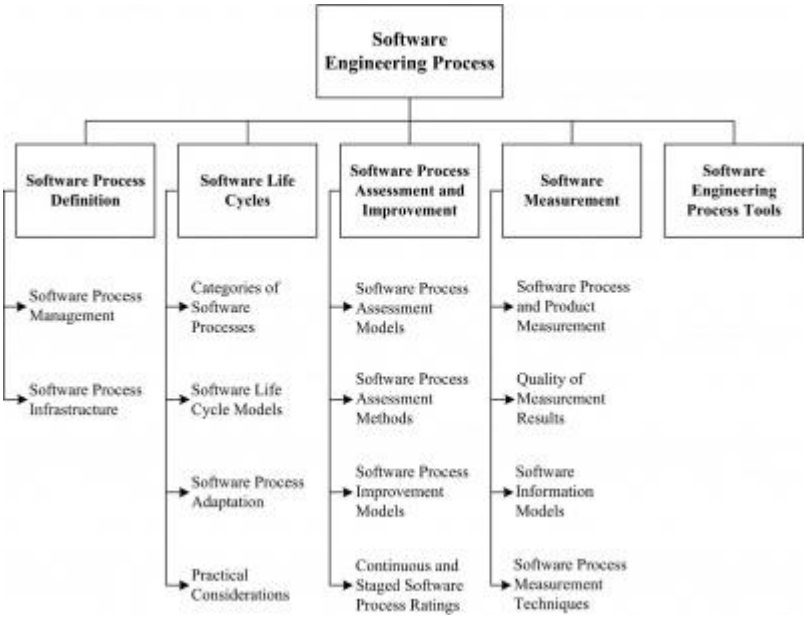


Figure 2

One of the greatest problems facing the software engineering process is how to accurately measure the process. This issue has been the subject of numerous academic articles, as well as being a frequent issue in the real - world software engineering industry. Exactly how the process is measured is discussed in this report.

# 2. Measurable Data

This section will examine measuring data, what this means in software engineering and the purpose of the measurements.

Figure 3

## 2.1. Why do we Measure Productivity?

The software engineering process is, by nature, difficult to measure. The purpose of the industry is essentially to solve problems using software, so as of yet a definitive general method of accurately measuring the productivity of any given software development team has eluded the software engineering industry. When the idea of measuring productivity was in the early stages of development, it was met with a good deal of resistance from developers, who believed that their time was better spent developing their software, rather than spending time analysing work that was already completed.

The software engineering industry has recently begun to warm to the idea of effectively measuring the productivity of the software engineering process. This is largely due to the increased clarity of the goals of this analysis, as well as the improving efficiency with which this analysis can be carried out.

The specific goals of measuring productivity in a high-functioning industry are discussed by Kevin Anderson in “Organisational Linkages: Understanding the Productivity Paradox”, 1994:

*“In industry, the measurement and analysis of individual-level productivity serves the following five major functions:*

1. *Define Productivity and Direct Behaviour*
2. *Monitor performance and provide feedback*
3. *Diagnose Problems*
4. *Facilitate planning and control*
5. *Support innovation”* (Anderson, 1994)

Each of these functions are clearly applicable to the software engineering process, with function 5 perhaps being the most significant in an industry where innovation is key.

## 2.2. What Exactly are we Measuring?

As previously mentioned, quantifying aspects of software engineering to measure can be incredibly difficult, with so many potential problems that developers could solve. There are some simple possibilities, such as lines of code and run-time, but these are widely considered to be poor methods of measuring the software engineering process. Bill Gates sums up his dislike of such methods in his quote:

*"Measuring programming progress by lines of code is like measuring aircraft building progress by weight."*

Gates argues that this should not be a general method for measuring the productivity of all software engineers. His reluctance to use such a measurement is understandable, with longer solutions sometimes providing a more complete solution than shorter ones, while shorter solutions can often run more efficiently.

In order to combat a single-minded approach to analysing software development, two different types of data are measured: product metrics and process metrics. Product metrics are concerned with the final product e.g. The length of code, the complexity of solution and it's reusability. Process metrics look at the product while in development, for example the number of commits to Git or the time taken to edit the code.

Any one of these metrics on their own would say very little about a project or development team, but when combined and compared against countless data from the past, they can give a relatively accurate picture of how the project / team is performing overall.

One measurement is mentioned by project managers time and time again, and yet is difficult to quantify and only becomes clear towards the end of the software development process – customer satisfaction [2]. If the customer is satisfied, then the solution itself becomes almost irrelevant. Every metric is essentially attempting to analyse the data to achieve this goal in the most efficient way possible

# 3. Computational Platforms

This section will discuss the computational platforms that are available to collect and analyse the metrics from performance during the software development process.

## 3.1. What are Computational Platforms?

Computational Platforms are used in the collection and analysis of physical data. This analysis must be carried out on the physical (code etc) data collected as running analysis on more abstract data is simply infeasible.

## 3.2. "The Streetlight Effect"

The data, subsequently, must be carefully chosen – as is discussed in Phillip. M. Johnson's paper "Searching Under the Streetlight for Useful Software Analytics". This paper discusses a common form of observational bias, often known as "The Streetlight Effect" [6], which is based on the following joke:

*"A police officer sees a drunken man intently searching the ground near a lamppost and asks him the goal of his quest. The inebriate replies that he is looking for his car keys, and the officer helps for a few minutes without success then he asks whether the man is certain that he dropped the keys near the lamppost.*

*"No," is the reply, "I lost the keys somewhere across the street." "Why look here?" asks the surprised and irritated officer. "The light is much better here," the intoxicated man responds with aplomb."* [7]

"The Streetlight Effect" details what happens when analysis is done based on ease of access, rather than measuring data that will give a true representation of a project's success. Returning to a previous example, if a team's productivity is based solely on the number of lines of code due to the ease with which one can obtain this data, it is likely that many other issues /

successes of the project will fall through the cracks and fail to be reported in the analytics report. Choosing what data to analyse can be difficult, so many companies now offer services and software that both choose the data to be analysed and carry out the analysis on it. This relieves pressure on the management team of the development company, who may not have a wealth of experience in the data analytics industry.

## 3.3. Breakdown of Computational Platforms

There are a number of different computational platforms that can be used to analyse data from the software development process. The following are just a few of the platforms available.

### 3.3.1 Personal Software Process (PSP)

*"The Personal Software Process (PSP) provides engineers with a disciplined personal framework for doing software work. The PSP process consists of a set of methods, forms, and scripts that show software engineers how to plan, measure, and manage their work."*

(Humphry, 2000). This platform essentially looks at ways for individual software engineers to improve their productivity throughout the development process. It provides them with the tools necessary to plan out and execute a project efficiently.

| Advantages | Disadvantages |
|---|---|
| Each engineer can base their plan and execution their specific problem | PSP must be scaled to different sizes based on the current project – this can be challenging. |
| Cost effective as PSP helps prevent defects from occurring – prevention is cheaper than removal. | Checking the validity of the observations can be difficult – however they must be validated for the analysis to carry weight. |
| Defects are often caught earlier in the development process – saves money. | It is a largely manual process – Human Error can be a major issue in validity of results. |

[8] [9]

### 3.3.2 HackyStat

*"Hackystat is an open source framework for collection, analysis, visualization, interpretation, annotation, and dissemination of software development process and product data."* [10].

Hackystat is a product of the evolution of open-source communities. It is used to collect information that is *"useful for quality assurance, project planning, and resource management"* [10].

The program provides an easy-to-read, comprehensive table as shown below. It is quick and easy to see the values for vital data such as code coverage, complexity of solution and number of commits

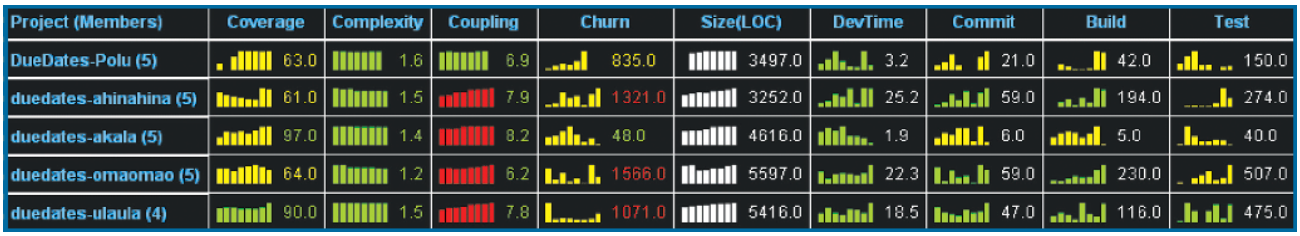


Figure 4 – A visualisation of Hackystat [6]

| Advantages | Disadvantages |
|---|---|
| Hackystat is completely automated - no human error involved in collection and analysis of data. | Primary use is in academia – has yet to make a significant breakthrough into industry. |
| Better than other platforms for more complicated analysis. | Hackstat's architecture is "language-independent" – majority of sensor are only made to collect data from Java-based systems. |

[11] [12]

### 3.3.3 Leap Toolkit

Leap Toolkit is a partially manual input system which was designed to overcome some of the flaws of the PSP system, namely the data quality issues. The use of the Leap Toolkit made it clear that it would be impossible to completely automate PSP, which in turn led to the creation of several different automated platforms.

| Advantages | Disadvantages |
|---|---|
| Provides more analysis on data that PSP, including regression analysis | Human error still exists with the manual inputs |

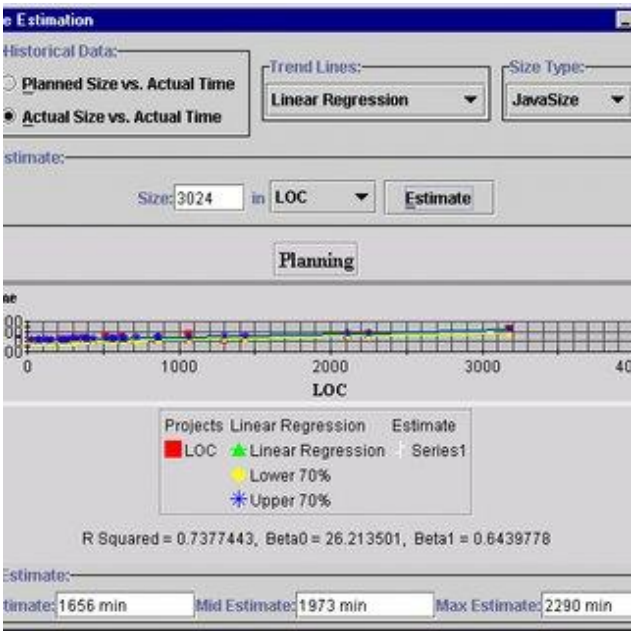


Figure 5 – A visualisation of the Leap Toolkit

There also exist many other platforms for analyses of such data, the majority of which have been created in recent years by specialised software analytics companies. Some examples of platforms not discussed in this report are Prom, Sonar and Ohloh, but numerous other successful platforms are available.

# 4. Algorithmic Approaches

This section will discuss the algorithmic approaches to solving the issues involved with measuring the software development process (as is seen in the human error or PSP and the Leap Toolkit). The discussion will focus on machine learning methods, both supervised and unsupervised, but I will discuss another non-machine learning method (PQI).

## 4.1. What is Machine Learning?

*"Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience"* [13]. Essentially, a computer is *"learning"* what to do in a given situation from previous experiences. This is an area of great interest to software engineers nowadays, and one of the areas where it could be applied in the future is in the analyses of the software development process.

There are two main types of machine learning: supervised and unsupervised. This report will discuss methods from both types.

## 4.2. Supervised Machine Learning

This method of machine learning is currently more popular in the software engineering community. The following is an explanation of supervised machine learning:

*"Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output. Y = f(X)"* [14]

The ultimate goal is to be able to predict the outcome Y given an input variable X. This is done by mapping input variables to output variables until there is a clear enough picture to make an accurate prediction of Y for a new input variable X. A specific supervised algorithm is discussed below.

### 4.2.1. K-Nearest Neighbours

K-Nearest Neighbours is a non-parametric algorithm. There can be several "classes" of data and we wish to classify to which of these classes a new input variable belongs.

KNN doesn't make any assumptions about data in any class. It assumes that similar data points usually lie close together. A new data point that we wish to predict the outcome for is passed to the algorithm. The algorithm creates a set by finding the K (chosen prior to analysis) nearest points to that data point on the graph. The new data point is classified as the class which contains the most points in this new set.

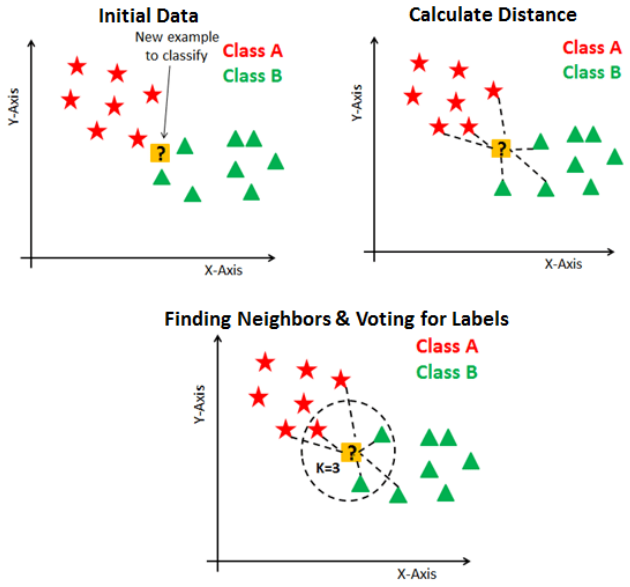


Figure 6 – K-Nearest-Neighbours Algorithm

| Advantages | Disadvantages |
|---|---|
| It is very simple to implement | Determining the value of k can be a challenge |
| It is robust to noisy data | Determining what values for distance to use can be tough |
| It is useful for large amounts of data | Has a high computing cost |

### 4.2.2. Bayesian Belief Networks

A Bayesian Belief Network is essentially a simple way to use Bayes Theorem in a complex dataset. It is used to combat the imperfect metrics found in other methods.

The Bayesian Belief Network model is a graphical network with an linked set of probability tables. The algorithm computes a probabilistic value for each variable in the model, regardless of the degree of uncertainty linked to the variable. This probabilistic

value is computed through empirical analysis. The model collects a wide range of data about the process' efficiency throughout building and testing

| Advantages | Disadvantages |
|---|---|
| Accounts for uncertainty and incomplete information | Tougher to implement than other algorithms |
| Able to produce results for all comparisons of interest within a connected network | Has not been tested in certain scenarios |

## 4.3. Unsupervised Machine Learning

Unsupervised Machine Learning has an input variable but no set output result. There aren't necessarily any correct or incorrect results, so unsupervised machine learning involves manipulating the underlying structure of the data to find a result. An example of this, K-Means Clustering, is discussed below. [14]

### 4.3.1. K-Means Clustering

The basis of K-Means Clustering is to break the data into a series of groups, or clusters. The data is broken into a fixed number (k) of clusters. The algorithm creates k "centroids", with each centroid placed far apart where it is believed cluster will form to ensure that the clusters that will form are correct.

The algorithm passes through the data and finds which centroid each data point is closest to on the graph. The data point is then assigned to that centroid to help form the cluster. Once all points have been classified, the algorithm takes an average of the points in a cluster and assigns a new centroid for the cluster at that point. Below is an example of K-Means clustering for k = 3. The final centroid positions are not shown in this visualisation.

| Advantages | Disadvantages |
|---|---|
| Easy to implement | Like K-Nearest Neighbours, finding a value for k can be difficult |
| Can produce tighter clusters than other clustering methods | The final results can be influenced by the order the data is inputted |

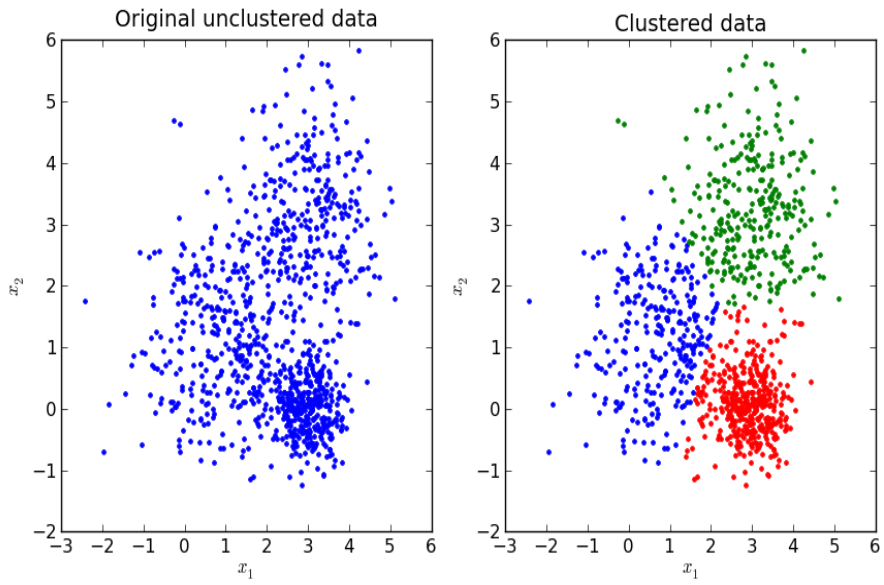


Figure 7 – K-Means Clustering

# 5. Ethics

A key part of analysing the software development process is the ethics involved. At the end of the day, there are human beings behind the screen and over-analysing them could have a severely detrimental effect on both their mental health and their work.

This section will discuss a number of real-world examples of when data analytics has gone too far. The facts will be detailed, and I will be giving my personal opinion on the matters.

## 5.1. Cambridge Analytica and Facebook

Cambridge Analytica is a political data analytics firm. While not directly linked to the software engineering process, the example of how they misused private data from millions of individuals shows the damage that data analytics can cause if collected or analysed in the wrong way.

Cambridge Analytica's unique selling point was that they had a very large amount of data on people, primarily in America, which allowed them to accurately target ads to specific groups of people. At the moment, this sounds very similar to what other analytics firms do, which isn't necessarily a bad thing. The problem arises when one looks at how their data was obtained. They are accused of secretly obtaining and using data from Facebook on millions of Americans without their consent. This data was then used to provide targeted ads for Donald Trump's 2016 presidential campaign. These ads were a large part of Trump's campaign, and it was

later revealed that these ads were specifically targeted using data that had been obtained unethically.

I believe that breaches of privacy of this kind are completely unacceptable. The data itself was obtained without Facebook users' permission, so both Facebook and Cambridge Analytica are at fault. Facebook should have protocols in place that ensure users' data is secure, and they certainly should not have a hand in supplying this information to outside firms. Cambridge Analytica should never have used data obtained in such a fashion to target ads to Facebook users', potentially compromising the integrity of the US presidential election. [15]

## 5.2. GDPR

GDPR or General Data Protection Regulation, is a set of data protection regulations put in place by the European Union in 2018. On a basic level, GDPR is designed to give EU citizens more control of their personal data, essentially to prevent situations like the Cambridge Analytica situation discussed above. GDPR provides an up-to-date set of laws for the internet era that all businesses operating in the EU must follow. These laws ensure that companies obtain data legally and they have the obligation to protect that data from misuse.

I see these new laws as a step in the right direction for data protection. I believe that every person should have control over their personal information, and that the collection and use of this data without consent is a severe breach of privacy and completely unethical. The next step is for there to be worldwide legislation on data protection, and this legislation will need to be added to as technology develops around us.

## 5.3. The Software Engineering Process

While these examples aren't directly related to the software engineering process, they provide a good outline for practises I believe should be in place when analysing the development process. A situation will never arise where a developer's personal information should be obtained by a data analytics or software engineering firm. This, along with over-analysing the process, is where I draw the line between ethical and unethical practises.

I briefly discussed the affect that over-analysing a worker can have on them. It is common for workers to become less motivated and more stressed if they are over-analysed. It seems clear to me that a worker should not be analysed during their break, and I think that analysing workers on things like what other employees they are talking to during the day is a step too far. Analysis like this has the potential to create a hostile work environment and could cause productivity to lessen and teamwork to become less effective.

# 6. Conclusion

The software engineering process is a difficult process to optimise. It is tough to analyse effectively, and hence improvements to make aren't always clear. There are some successful platforms and algorithms available to help with the analysing process, yet no clear method has been found to effectively analyse all development processes, due to the varied nature of projects. However, firms must be careful when trying to improve their data analytics, as there is a human aspect involved in the software engineering process. They must ensure to comply by laws such as those detailed in GDPR, as well as ensuring they don't over-analyse their employees.

# Bibliography

[1] Anderson, P. 2018. “How do I measure the productivity of my software development team?” [Online]

Available at: https://tasktop.com/blog/measure-productivity-software-development-team/

[2] Stackify. 2017. “Development Leaders Reveal the Best Metrics for Measuring Software Development Productivity” [Online]

Available at: https://stackify.com/measuring-software-development-productivity/

[3] White Paper “Understanding Software Development Productivity from the Ground Up” [Online]

[4] Ruch, W. A. 1994. “Organizational Linkages: Understanding the Productivity Paradox” [Online]

Available at: https://www.nap.edu/read/2135/chapter/6

[5] Sillitti, A. Et Al. 2003. “Collecting, integrating and analyzing software metrics and personal software”

[6] Johnson, P. M., 2013 “Searching under the Streetlight for Useful Software Analytics”

[7] Quote Investigator. 2013. ““Did You Lose the Keys Here?” “No, But the Light Is Much Better Here”” [Online]

Available at: https://quoteinvestigator.com/2013/04/11/better-light/

[8] Humphry, W. S. 2000. “The Personal Software Process (PSP)” [Online]

Available at: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5283

[9] Khan, A. K., 2012. “Impact of Personal Software Process on Software Quality” [Online]

Available at:
https://pdfs.semanticscholar.org/e97d/2611d22c5ee8f8b70c0dc00bff626c54d324.pdf

[10] “HackyStat” [Online]

Available at: https://hackystat.github.io/

[11] Johnson, P., 2001. “Project Hackystat: Accelerating adoption of empirically guided software development through non-disruptive, developer-centric, in-process data collection and analysis” [Online]

Available at: https://pdfs.semanticscholar.org/33d3/41507e26eb3b9b43bc99b53952d4ed3f6aa5.pdf

[12] Stack Overflow., 2010. “ Is Hackystat only academically interesting?” [Online]

Available at: https://stackoverflow.com/questions/2028213/is-hackystat-only-academically-interesting

[13] Mitchell, T., 1997. “Machine Learning”

[14] Brownlee, J. “Supervised and Unsupervised Machine Learning Algorithms” [Online]

Available at: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[15] Wired. “The Cambridge Analytica Story, Explained” [Online]

Available at: https://www.wired.com/amp-stories/cambridge-analytica-explainer/

[16] Mitchell, T., 1997. “Machine Learning”

Figure 1 - https://www.niitkenya.com

Figure 2 - https://stackify.com

Figure 3 - http://swebokwiki.org/Chapter_8:_Software_Engineering_Process

Figure 5 - https://www.researchgate.net/figure/Leap-Toolkit-Controller-This-is-the-main-controller-for-the-Leap-toolkit-The_fig1_2647145

Figure 6 - https://www.kdnuggets.com/2019/07/classifying-heart-disease-using-k-nearest-neighbors.html

Figure 7 - https://mubaris.com/posts/kmeans-clustering/