CSU33012
Thomas Dixon
17341291

# Measuring Software Engineering


## Introduction

Productivity is a challenging concept to define, describe and to measure for any kind of software engineering. In measuring software engineering, we use software metrics. A software metric is a measure of software characteristics which are quantifiable or countable. We use software metrics for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses. Effectively measuring software engineering is a difficult and nuanced task. Originally software engineering was measured in the seemingly most obvious way, lines of code written. This turned out to be inaccurate and even counterproductive as software engineers soon began to manipulate this metric by stringing out their code to make it longer and in doing so, less efficient. Bill Gates is famously quoted to have said “Measuring programming progress by lines of code is like measuring aircraft building progress by weight.”

# How then can Software Engineering be measured?

In the early days of data analysis, one of the main difficulties was collecting enough data to analyse. In modern times this problem has been flipped on its head and we now have to traverse huge volumes of data and extract which data is relevant to our analysis. The problem then arises of which metrics are most effective when measuring software engineering and how we can manipulate our reams of data to best illustrate this.

There is no metric that is necessarily better than another, it really depends on what the goals are for the software development team involved. Some examples of such metrics are:

**Lead time**

Lead time measures how long it takes for ideas to be developed and delivered as software. Lowering lead time is a way to improve how responsive software developers are to customers.

**Cycle time**

Cycle time describes how long it takes to change the software system and implement that change in production.

**Team velocity**

Team velocity measures how many software units a team completes in an iteration or sprint. This is an internal metric that should not be used to compare software development teams. The definition of deliverables changes for individual software development teams over time and the definitions are different for different teams.

**Coupling**

This is a measure of the degree to which different classes are related. Different classes should be independent of one another. The higher the coupling measurement, the harder it will be to change something in one class without affecting another. Developers should look to keep a low coupling value, this will

make classes more independent and easier to change something in the class without causing much conflict.

### Coverage

Indicates the code that is covered when tested. Generally measured as a percent, a developer will look to increase this. The higher this value is, the lower the chances are of having undetected software bugs compared to a program with low code coverage.

### Active days

Active days is a measure of how much time a software developer contributes code to the software development project. This does not include planning and administrative tasks. Can also see if it is progressive, they are starting early or leaving it all to the last minute.

### Efficiency

Efficiency attempts to measure the amount of productive code contributed by a software developer. The amount of churn shows the lack of productive code. Thus a software developer with a low churn could have highly efficient code.

### Code churn

This measures the amount of code that the developer has rewritten within a short time. If code churn increases, then it could be a sign that the software development project needs attention.

### Size-oriented metrics

Size-oriented metrics focus on the size of the software and are usually expressed as kilo lines of code (KLOC). Once a standard has been agreed upon for what constitutes a line of code, this is a simple metric to calculate. Unfortunately, it is not useful for comparing software projects written in different languages and also does not take into account inefficient code and duplication. One such way to measure its efficiency is errors per KLOC.

**Discussion on the use of these metrics:**

These metrics are not a proven science and there is doubt over their efficacy and even worry about some negative effects they could have on real human beings in the workplace. Just like software engineers found ways of manipulating measurement based on lines of code, they no doubt find ways to manipulate the findings of these other metrics. There is also the potential for negative effects of employees being under such constant scrutiny. These could include increased stress levels and damage to inter-staff relationships as a result of competition and judgement based off of these metrics. It may force some people to work in a way which doesn't work for them/suit them so as to achieve favourable results from these metrics. We must also consider the question of how far is 'too far' when it comes to monitoring and gathering of data on employees. An argument can be made of an invasion of privacy with some of the metrics listed above. On the flip side, rewarding good habits and a strong work ethic from those who are shown to do well from these metrics could encourage hard work and healthy competition from the employees. One could also find that through these metrics software engineers can be self reflective, and learn from their own experience what works for them and what they could do to improve. The most important thing to remember is that software engineers are people and not machines and that there is no one size fits all when it comes to what works for people and what doesn't. Management and leadership are of utmost importance in this regard. The results from these metrics should be used to help ensure that an employee's standing is based on their work and contribution in the workplace rather than how they may come across to those in charge.

# What platforms can be used to gather and process data?

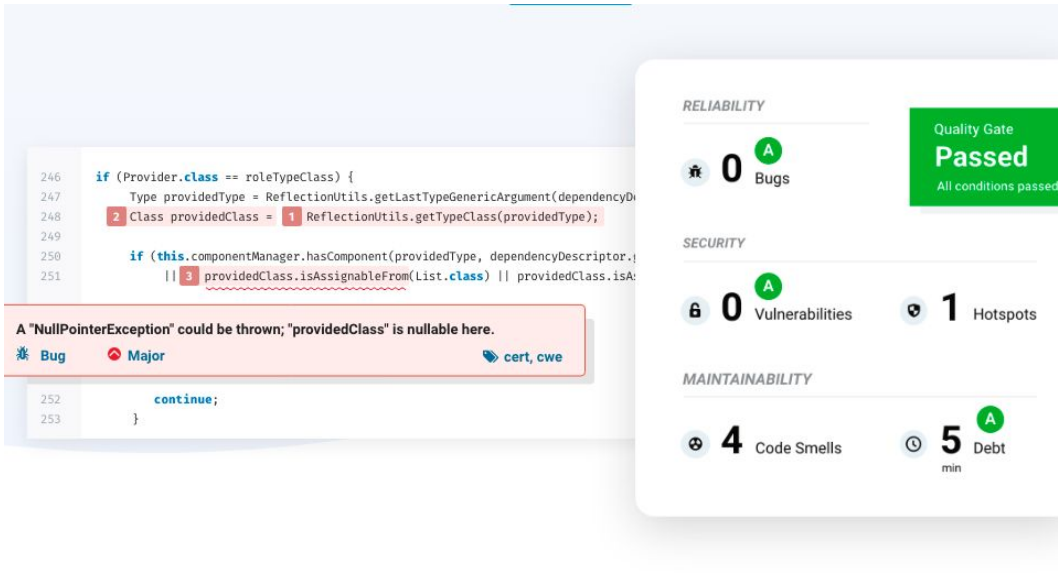
Personal Software Processes(PSP) was developed by the Software Engineering Institute(SEI) to look at the possible improvements of individual software engineers. A software engineer, named Watts Humphrey, wrote a book that detailed some aspects of the PSP. It mainly focused on a manual input process. In the book were many spreadsheet templates that had a range of inputs that needed to be recorded throughout a developer's work. It would be looking for log times and estimations made by the developer, along with some errors found. The process requires personal judgement for the developer, such as logging interruptions to their work throughout the day. Like any manual human input, there is human error or subconscious bias which can hinder analytics and data quality. Research for solving this problem led to the development of platforms such as servicenow.

### Servicenow

The servicenow toolkit looked at some of the quality problems found in PSP and addressed them by automating and normalizing the data analysis process. This new technique still requires some manual input of data from the developer but the machine will produce some analysis that the PSP would not provide. It will perform various forms of regression analysis on the data. Between the manual input and automated calculations of the software. Some of the functionality of servicenow is to check the size of the work product; the time it takes to develop it; the defects that the user or others find in it; the patterns that they discover during the development of it; checklists that they use during or design as a result of the project; estimates for time or size that they generate during the project; and the goals, questions, and measures that the user uses to motivate the data recording.

## Sonarqube

Sonarqube is an open source platform used to manage the quality of source code. It is designed to manage code quality with minimal effort to the user. It can be used for once-off audits throughout development of projects. Sonarqube measures and monitors variables such as duplicated code, coding standards, unit tests, complex code, potential bugs and comments. Sonarqube was originally developed for Java but with the help of plugins can adopt others like Flex, PHP, PL/SQL and Cobol. It has multiple products depending on the service required. It can help companies manage technical debt issues and can display error descriptions to help developers detect them.



## Pluralsight - Flow

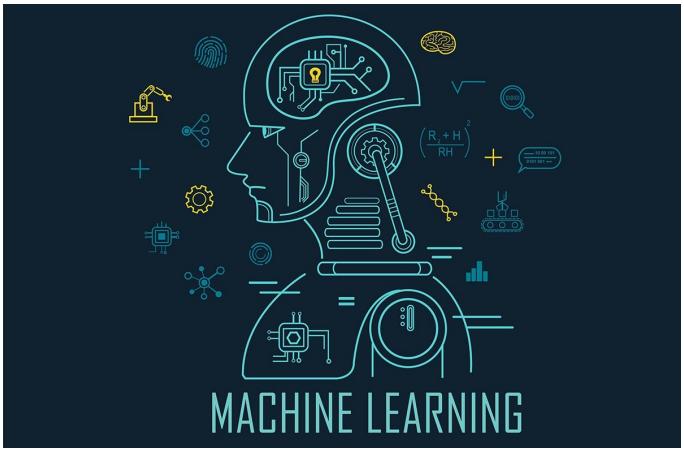
Flow gives engineering teams access to the same level of metrics and concrete data other departments in the organization have relied on for years. It analyzes data from git repositories, code reviews and issue trackers, and packages them in easy to understand reports. With the ability to visualize work patterns engineering leaders can debug their development process with data to help their teams have a more efficient workflow. In addition to traditional metrics, Flow uses git analytics from the github API to measure and supply metrics on collaboration and interaction between engineers such as responsiveness of submitters to comments on their pull requests and to what percentage of comments a submitter responds. These metrics can be used to prompt and encourage healthy discussion in reviews.

# What algorithms can we use to analyse this data?

A key part of measuring software engineering is the human analysis of human behaviour and human error, in saying this it makes sense that there would similarly be human error and bias in the analysis of this behaviour. This is where algorithms and analysis technology comes in. Through the use of the historical input and efforts made, a machine can be programmed to read off these and predict results. This new development is known as machine learning. The machine will be given learning algorithms of fault prone and non fault prone methods to develop predictive models of quality. There are mainly three types of machine learning algorithms but there are many machine learning algorithms under these categories, and as software engineering advances more will be produced.

## Machine learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.



There are three main types of machine learning:

## Supervised Machine Learning:

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. This is the most popular technique in machine learning. As an example, given an input of variables and an expected output such as 'y = f(x)'. The expected output is called the supervisory signal. The idea is not to only suit the current 'x' inputs but to allow for varying 'x' inputs in the future that will conclude a respective response 'y' output, where 'y' is a function of 'x'. It is called supervised because we know the answer expected. While the algorithm is making its predictions on the input data it is being monitored and corrected where needs be. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Eg. Support Vector Machines-

In machine learning, support-vector machines (SVMs) are supervised learning models. Given a set of training data, each assigned to one of two categories. An SVM training algorithm builds a model that picks which category new data belongs in. When we model this data it is represented as points in space, with the points in the separate categories divided by a gap. New examples are added to this model and predicted to belong to a category based on the side of the gap on which they fall.

## Unsupervised Machine Learning

With unsupervised machine learning we build a set of input variables but no corresponding output variable expected. This models the underlying structure in the data so that it can be more understood. There is no constant monitoring and no attempts to adjust depending. From this algorithm, the output we are looking for is based on commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. They can be grouped by clustering and associations. Clustering groups the variables with similarity and dissimilarity behaviours.

Eg Apriori Algorithm

The Apriori algorithm is an algorithm for frequent item set mining and association rule learning over relational databases. The parameters "support" and "confidence" are used. Support refers to items' frequency of occurrence; confidence is a conditional probability. Items in a transaction form an item set. The algorithm begins by identifying frequent, individual items (items with a frequency greater than

or equal to the given support) in the database and continues to extend them to larger, frequent itemsets.

### Reinforcement learning

Reinforcement learning is the training of machine learning models to make a sequence of decisions. The aim is for the AI to achieve a goal in the long term in a potentially complex environment. In reinforcement learning, AI faces a game-like situation. The AI uses a technique of trial and error. To get the machine to do what the programmer wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward.

Although the designer sets the reward policy–that is, the rules of the game–he gives the model no hints or suggestions for how to solve the game.

Eg Markov Decision Process (MDP) is a mathematical framework used for modeling decision-making problems where the outcomes are partly random and partly controllable.

## Is this ethical?


Ethics are moral principles that an individual must apply when using their skills. They should use their professionalism to make decisions with regards to professional and social conduct. There is a code of ethics that software engineers abide by, they are approved by the ACM and the IEEE-CS as a competent standard for learning and current software engineer professionals. Much of the ethics for a software engineer can be considered universal to many roles. Many of the main principles are understanding what is good practice and what is not. Not abiding by this code of ethics, can tarnish a professionals career.

The idea of ethics is much broader than the specific area that we are interested in, it applies to all areas of life. What is right and wrong. Humans first attempt at writing a code of ethics can be seen in religious texts. In this paper we are concerned with ethics in computing. What makes this so interesting is that because of constant developments in technology, we must constantly be coming up with new regulations and ideas of what is right and wrong. In 1950 the world's first computer crime was committed. A programmer was able to use a bit of computer code to stop his banking account from being flagged as overdrawn. However, there were no laws in place at that time to stop him, and as a result he was not charged. To make sure another person did not follow suit, an ethics code for computers was

needed. We find ourselves in this position in recent times with advancements in technology such as artificial intelligence we must constantly be evolving our laws and ethics to adapt to this new world.

For years there has been research done to try and improve software development, not only in the sense of code writing but as reliability and professional approaches to development stages. Like any profession, process or procedure, if there is room for improvement it is ideal to do so. When looking to make these improvements it goes without saying that there must be a way to gauge and measure their efficacy. These metrics require comprehensive historical data. This is where conflict arises between the developers and management. Software developers are concerned about the monitoring of their behaviour on work and their work efforts. Many argue that this monitoring has gone too far and is against moral status and intrudes on their rights. This area is rapidly evolving and I feel is running a fine line between right and wrong in many new areas of advancement. Some platforms require manually calculated input data, and others are automated. The manual input is more flexible to the developers' judgement, whereas automated platforms read directly from the computer and monitor their stages at each moment in time and report statistics with respect to the data analysed.

It is the automated data collection that worries me in terms of ethics, what exactly constitutes an invasion of privacy for instance. With new technology now monitoring every keystroke developers make and tracking their physical movements from their chair and desk to the amount of hours spent at each specific project. To me this feels like it is dehumanizing the engineers and almost treating them like livestock. In recent months with people working from home, there have been reports of management reserving the right to turn on employees webcams at any time without their knowledge to check whether they are indeed working from home. This is absurd and crosses the line, a line which we must fight to keep in place in coming years as further advancements in this kind of technology emerge.

[1](n.d.). What is Flow exactly? - Pluralsight Help Center. Retrieved November 20, 2020, from https://help.pluralsight.com/help/what-is-flow-exactly
[2](n.d.). 9 metrics that can make a difference to today's software .... Retrieved November 20, 2020, from https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams

[3](n.d.). International Organization for .... Retrieved November 21, 2020, from https://en.wikipedia.org/wiki/International_Organization_for_Standardization

[4](2017, September 16). What Are Software Metrics and How Can You Track ... - Stackify. Retrieved November 21, 2020, from https://stackify.com/track-software-metrics/

[5](2020, March 18). Software Measurement and Metrics - GeeksforGeeks. Retrieved November 24, 2020, from https://www.geeksforgeeks.org/software-measurement-and-metrics/

[6] (n.d.). Ten Commandments of Computer Ethics - Wikipedia. Retrieved November 25, 2020, from https://en.wikipedia.org/wiki/Ten_Commandments_of_Computer_Ethics