Measuring Engineering

Alex Smirnov

Student No: 15322057

Introduction

To this day there is not a universal method or approach to analyze and measure pieces of software. As every project is different to the next, there are different types of measuring that might be better for one project then another, as well as different developers having preference to one.

In this report I wish to look at a number of methods that are used to analyze and measure the software engineering process, including things like measurable data present, the computational platforms used to gather and analyze data, the algorithmic approaches to the analysis and the ethical concerns that could be brought up with their use. I will also look at how useful and impactful these tools are to the improvement of the development process.

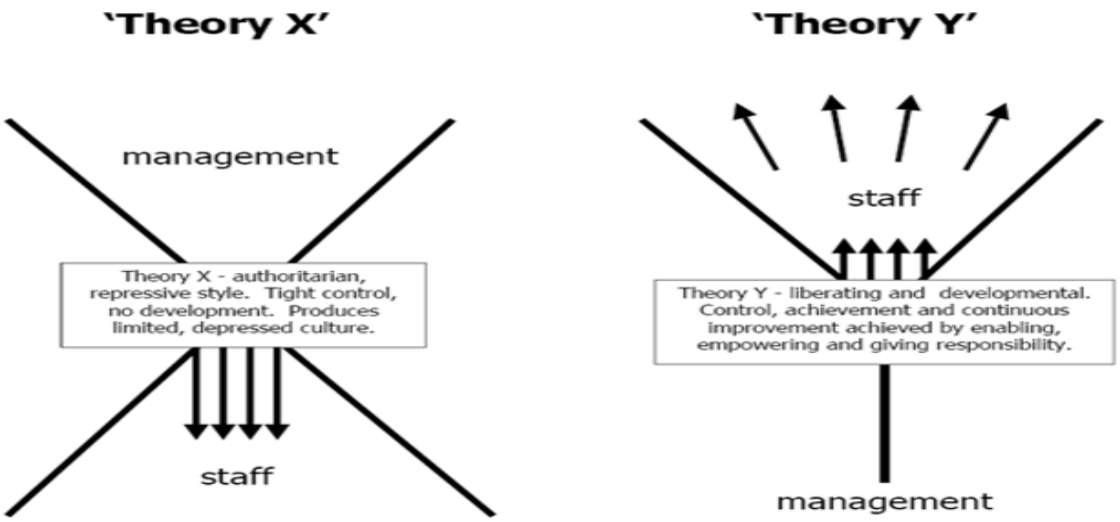
Measuring Data

Measuring software engineering in the submission of forms is very like the way general management is performed. “Taylorism” is the name given to a set of principles in the development of scientific management. With “Taylorism” the aim is to improve predictability, efficiency and control in the developing factories. These principles portray a negative image of the average employee, being seen as a lazy, untrustworthy, closely monitored lay about. These old principles were a strong influence to the management of its time and

can be seen implemented even in today's society with the Hackystat methodology. In this the programmer must be monitored and tabs must be kept on him/her as to make sure that they are on course and not straying from the task at hand. For this monitoring tools are installed in the programmers' machine so that management can be fully informed on what they are doing.

During the 1960s a new management philosophy was thought of by Douglas McGregor. He contrasted "Taylorism" (Theory X), which assumes that workers are lazy and not bothered preforming on their own initiative, to what he calls Theory Y, which assumes that workers are motivated and willing to work if the correct support is given by the management. This type of management style is proven to be more popular and effective in the modern society and is seen prevalent today in the software development community. This allows the developer to make use of their time not focusing on a strict input/output style of working but on their own initiative and produce work that they can feel happy doing. As McGregor said:

"Management cannot provide a man with self-respect or with the respect of his fellows or with the satisfaction of needs for self-fulfillment. It can create conditions such that he is encouraged and enabled to seek such satisfactions for himself, or it can thwart him by failing to create those conditions."



(McGregors "Theory X" vs "Theory Y")

An alternate simplistic solution to measuring data could be to measure the lines of code written by the software engineer. This is a simplistic yet deeply flawed way of measurement, as the engineer with the most lines of code is considered the best worker, regardless of the progress made in the project. The engineers are placed in a constant competitive environment to write the most amount of lines for a solution.

This method has some major flaws, one being that the engineer can intentionally needlessly extend his code so that it is larger than it needs to be. Examples of this could be extra spaces separating lines, formatting statements in a way to increase the lines. As software engineers, having the least amount of code to perform a function is often desired. Hence penalizing a worker for completing a task using significantly less lines of code doesn't make sense, and usually the solution is more elegant and thought out. Because of this another method must be used.

Code is always being edited and changed over time. Because of this there needs to be a way of measuring the effectiveness of the

additions made to the code. Code churn is a measureable piece of data that can be used to notice errors in the system. Code churn measures the change of a section of code over time, meaning that for instance high code churns occurs when a section of code is being frequently modified. If there is frequent modifications to the code, the code churn is able to spot if the additions are positive or negative ones, making it easy to spot out workers wasting time.

Often times two software engineers may come back with two pieces of code that preform the same function correctly. In this case, how do we distinguish these two codes and how do we choose which one is more efficient and is to be used? A form of measurement that allows us to do this is cyclomatic complexity, which measures the complexity of code by analyzing how many paths can be possibly taken in the code. The basic idea of it is to see if a solution is overly complicated. This means that the piece of code with the least complication is generally seen as the better piece of code. It is seen that the more complicated piece of code is more susceptible to bugs and other undesirable results.

Another way of measuring data is code coverage. This is used to see how much of the selected program is run when the testing is carried out. The ideal situation is that all the lines are executed when testing. This can be used to assess if the software engineer has thoroughly written and tested the code they have written. This way we can see if the software engineer is slacking or not devoting their full attention. The problem with code coverage is that even if the code that is tested has high code coverage this doesn’t mean that it is good code. The tests that are run to the code are very important in actually testing the code, so not only should there be high code coverage, there should be useful and effective tests run to ensure

bug free code. There is a problem with the engineer being able to write tests that cover code and pass, but actually do little to test the code. This is a major problem with code coverage.

Computational Platforms

There are a variety of platforms available to the public for measuring of data. The basic goal of these open source and enterprise tools is to collect data and display it to whoever is interested.

Like mentioned above, Hackystat is a computational platform used to collect data on a software engineer. Hackystat has access to the engineers IDEs and any build environments they use. With this they are able to store information on the work done and save it to the Hackystat server. This information can be used to compare with other engineers, allowing management to assess the work done by the engineers. The information obtained this way can help with planning a software project as they know the speed of which work is getting done and are able to see if any engineers are performing poorly in the project.

Repositories like GitLab and GitHub are not only there to allow collaboration and back up data. It can also be used as platforms to collect data on software projects. GitHub for example is free to use, unless private unlimited repositories are required. The advantage of these repos is that they provide the user with data in the form of graphs or stats for each commit. GitHub can also gather and display the lines of code that were written by what developer, when someone has made a change in the code, which lines where edited

and how frequent a contributor is committing to the project. Things like are vital to a team coordinator, as quality and workflow can be easily displayed and measured.

(Example of GitHub graph: User Contribution)

IDEs such as Visual Studio also facilitate for testing. Visual Studio has a tool which measures code coverage for each function. This way there is direct feedback given in real time. There are also a number of plugins available for things such as unit testing, which supports the popular programming languages.

Algorithmic Approach

There are many forms of algorithmic approaches that allow calculation for the data being measured.

Code churn has a very simple calculation algorithm. Calculating code churn on a platform like GitHub is as easy as adding up the added, modified and deleted lines for a certain function or commit. These values are easy to obtain and represent easy to understand information.

The algorithm to calculate the cyclomatic complexity of a program is very simple for programs that do not use any sort of branch statement. We can set a counter to one and increment it for every time we enter a for, if, while loop. Thomas McCabe is the man responsible for cyclomatic complexity, has a mathematical definition defined in his book on the topic as “cyclomatic complexity v(G) of a graph G with n vertices, e edges and p connected components is:

$$v(G) = e – n + p”.$$

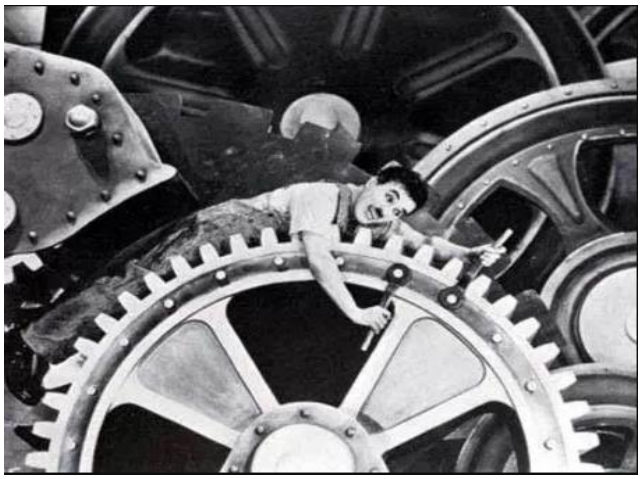
Other algorithmic approaches such as bug reports, code comments and mining data can be used to build up a good view of the project.

Ethical Concerns

As with anything, there are a number of ethical concerns which have to be considered when analyzing a working software engineer and measuring their work. These concerns must be addressed at the beginning of every project, no matter the size of the team or the magnitude of the project. There must be a strict code emplaced to protect the rights and wellbeing of the workers, for the company’s sake as happy workers are more productive (as seen above) and motivated. Nobody wants to work for someone who doesn’t respect their rights to privacy, so it is in the company’s best interest to address concerns like this to maintain their workforce.

Methods like the recording the screen of a worker and others closely resembling the “Taylorism” way of measuring performance, where the employee is closely monitored, in this case their computers are recorded, carries numerous ethical concerns. This is a big infringement on the privacy of a worker and is no different than a

“big brother” type figure constantly looking over your shoulder, creating a “1984” type workplace. This could be mentally draining for an engineer and can lead to a decrease in productivity. This type of measurement is unappealing to an engineer, who would be swayed from applying to such a company. Such measures should only be put in place if a worker is on a warning where being constantly monitored is required. Either way it is crucial that the worker is informed of this before starting work, unless the company want to find themselves in a lawsuit.



(Powerful image depicting the mental turmoil experienced by a “Tayloristic” management style)

A system such as Hackystat would be a less extreme method of monitoring a software engineer. This would monitor the editor and tools of the engineer, rather than the whole screen of the computer. This way a vast amount of information is being collected and the engineer has his privacy outside the work that he should be doing. This is less ethically concerning and if the worker is informed and agrees to these conditions, no problem should arise.

With a software engineer working for a company, he/she has given a lot of detail to the company about themselves and through years of work the vast amount of data stored on the employee could be

compromised if the wrong person had access to the files. If there was any sort of leak or breach this could seriously compromise the workers reputation and image. Because of this there are ethical concerns about how much and what type of information an engineer should be giving to the company they are employed by.

These ethical concerns should be discussed in detail with the software engineer before joining a company or before commencing on a new project as they are paramount to the protection and wellbeing of the worker.

Conclusion

To conclude, there are many methodologies present for measuring software engineering today. As we have learned however, there is no simple solution and every methodology has its own ups and downs and should be used accordantly given the situation. I believe that if agreed upon and not abused by the management, Hackystat is a good way for measuring data and progress made by the workers as there is a vast amount of data gathered this way and provides a way to monitor the project, insuring it stays on course.