Software Engineering CS3012

Measuring Engineering - A Report

Jonathan White - 17327237

whitej4@tcd.ie

Introduction

This report investigates and considers the ways in which software engineering as a process can be measured and analysed. It aims to address major questions facing today's software engineering industry, such as: what is engineering measurement? Why should we measure it in the first place? How can it be measured? And what are the implications of such an assessment?

With regards to measuring the software engineering process, this report will assess the measurable data, provide an overview of some of the currently available computational platforms which can perform such work, discuss the algorithmic approaches available, and will end in a discussion of the ethical concerns associated with this kind of analytics.

Software Engineering - A Definition

Software engineering is an engineering branch associated with the development of software products using well-defined scientific principles, methods and procedures [1]. Put simply, it is the application of engineering principles to the realm of software development. It is widely believed that a conference organised in the late 1960s by the NATO Science Affairs committee marked the official start of the profession of software engineering [2].

In the software engineering industry, with the majority of projects running over budget and over schedule, it has evolved into a profession concerned with how best to maximize the quality of software and the processes of creating it. This in turn requires companies and organisations to monitor and maintain a base of dedicated engineers who work productively and efficiently.

Engineering Measurement & Analysis

What is software engineering measurement, and what interest do software companies have in it? At the end of the day, the software industry is a cut throat industry with fierce competition and many new firms emerging daily. A company is only as successful and valuable as the products it builds, which are only as successful as its team of engineers. We do not live in a perfect world - in reality many engineers lack efficiency, are careless and lack motivation. It only takes one poor engineer on a team to cause issues and headaches for many others working on the same project. Therefore it is in a company's best interests to closely monitor and review the progress and output of their team of engineers. It's not just about the quantity of code, it's about quality as well.

Unlike other engineering industries and disciplines, measuring the work of a programmer is challenging and not as clear cut. However there are metrics available to quantify the value of an employee and the work they produce.

Measurable Data

What can we measure from a software engineer's work and how can we measure it? Below are some categories describing the metrics currently available, as well as some suggested ideas of what could be measured going into the future.

Lines of code

Measuring the lines of code (LOC) written by an individual has the advantage of being simple and easy to measure, and there is often a strong correlation between lines of code and effort [3]. However it is riddled with far too many flaws for it to be used as a main source of measurement. It encourages 'Copy-Paste-Syndrome', the use of meaningless comments and less efficient code. It often takes an experienced programmer fewer lines of code to complete a task, than an inexperienced programmer.

It has also been argued that LOC is a poor productivity metric because “it penalizes high-level languages: Assembler programmers produce five statements to a COBOL programmer’s one” [4]. Therefore this metric should not be heavily measured but may serve as a helpful indicator of productivity in some circumstances.

Number of commits

Similar to measuring LOC, commits can be a helpful metric to observe. However this measure can be abused. Some developers may take advantage by committing to a project repository every few minutes or each time after they’ve written only a couple lines of code. This metric also depends on a developer’s style of work: some developers like to commit after writing a simple function, some prefer to commit after writing several. It gives no indication of code quality. Inexperienced software engineers can often commit buggy code and then follow that commit with 10 more in an attempt to fix their mistake.

This metric, however, can be useful to observe the periods at which developers are working most actively on a project. Do their number of commits deteriorate after coming back from lunch break, or as they reach the end of the week. Do they commit less often when they are working from home? Monitoring the number of commits can be helpful in these circumstances.

Code coverage

If an engineer’s code has a high percentage of code coverage during testing, it suggests that it has a lower chance of containing undetected software bugs compared to a program with low test coverage [5]. Therefore measuring the successful coverage of a developer’s work can be a useful metric to measure their contributed value to a project. The downside is that developers may often refrain from making their code better, because it would negatively affect their coverage target.

Coupling and cohesion

Coupling is a good metric to measure the quality of an engineer's code. It is the degree of interdependence between software modules; a measure of how closely connected two routines or modules are [6]. Low coupling is often a sign of a well-engineered system. Therefore analyzing the independence of software modules is a helpful metric to consider, however doing this accurately may be a challenge.

Cohesion refers to what a class (or module) can do. A well engineered system component will have high cohesion, meaning it is focussed and specific in what it does. This is another helpful metric to consider, however it is challenging to measure this in an automated way.

Active development time

Measuring the time spent by a developer actively working on a project is a useful way of analysing their productivity. This could be achieved by adding an extension to their code editor to measure the time they spend actively programming, refactoring their code and testing their work.

However this does not take into account the time they spend verbally discussing architecture decisions with their colleagues, or the time they spend helping a junior engineer get to grips with a new technology. It doesn't monitor the logical reasoning and creativity the engineer has by sketching out concepts on a whiteboard by their desk. What if an engineer spends most of his day reading other people's code to understand the cause of a bug in the system? What if they spend the day in project meetings and informal discussions with colleagues who are having issues refactoring code to the company's new coding standard? This metric doesn't take such things into consideration.

Network usage logging

Using the internet is a fantastic asset to developers. As much as software engineers don't like to admit it, much of their time is spent on online forums like StackOverflow

or Quora when they run into problems. This is completely fine and should be expected from engineers of varying experience. However naturally with the internet only one click away from developers, it gives rise to much time wasting.
A method of measuring an engineer's productivity would be to monitor their network activity, whitelisting popular developer sites but recording when they sneakily visit instagram to check how many likes their latest post has, 5 times every hour. Of course, this should not be monitored excessively. Listening to music or a podcast in the background can often help developers focus while they work.

Code reviews

Project managers or supervisors can review the code of individual engineers by ensuring it complies to the coding standards set by the company. By reviewing their work, managers can analyse the quality of an individual's work by observing its readability, how well documented it is, and how efficient it is. The disadvantage is that it is time intensive and managers often do not have the time to manually review the code base of each of their subordinates.

Computational Platforms

There are a variety of tools and platforms available for measuring the software engineering process. Most involve cloud based analytics as a service - computation carried out on remote computers with a fee for the use of their services. Below is a summary of the most popular readily available services which companies can make use of to monitor the engineering efforts of their employees.

Code Climate

Code climate is a tool which ensures engineering teams only merge clear, maintainable, and well-tested code. Once you sign up, you can connect your repository and the system will analyze it. After reviewing all the code, it lets you know if there are code smells (characteristics in the source code of a program that possibly

indicates a deeper problem [7]), repeated code or security vulnerabilities. It will email you if any issues are detected.

Hackystat

Hackystat is an open source framework for collection, analysis, visualization, interpretation, annotation, and dissemination of software development process and product data [8]. It has the advantage of being free because it is open source, however it is not as well polished and lacks the versatility of other similar commercialised products. It should be noted that, as stated by its website, the tool is no longer under active development.

GitPrime

GitPrime has an impressive feature set. It has been described as giving a “360-degree view of the entire engineering process” and offers historical data, productivity graphs & reporting, codebase analysis and data visualisation. It aims to give you an idea of what "a day in the life" of your engineers is like and helps to identify project bottlenecks at a glance [9].

Humanyze

Tracking repository data is important, but as discussed previously in this report, an engineer’s productivity goes far beyond the code they write. Humanyze combines digital communications data with data from the physical world. It provides companies with “special badges that hang around your neck on a lanyard” [10] which carry out real-time analysis of your movement, activities and speech. They gather the employee data, process it and provide managers with dashboards displaying visual data and insights on employees.

Algorithmic Approaches

Once data has been obtained from measuring engineers with techniques outlined in the ‘Measurable Data’ section of this report, the next stage is to analyse the data to produce conclusions and helpful insights. Your measured data is only as valuable as

the insights and information it produces. Therefore the analysis stage is crucial for extracting meaning from the gathered data.

Computational Intelligence

Computational intelligence (CI) has been making waves in the world of data analytics. IEEE defines it as “the theory, design, application and development of biologically and linguistically motivated computational paradigms” [11]. It is the ability of a computer to learn how to carry out a specific task from observing data. CI encompases 3 main areas, one of which is Neural Networks.

Neural Networks and their potential use in measuring engineering

Artificial neural networks (NNs) are massive parallel distributed networks that have the ability to learn and generalize from examples [11]. They essentially are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns in data [12]. They are therefore able to establish correlations between present events and future events when trained with the correct data. This can be a tremendous advantage to companies looking to predict trends in the engineering process. For example, a company may use such a system to predict the development time or budget of a future project.

Regression Analysis

Regression analysis is a well-known and reliable method of identifying which variables have an impact on a topic of interest [13]. This comes in extremely useful when drawing conclusions from measured data. It can be used to answer questions such as: what factors contribute to on-time code deliveries? Does time of day affect the LOC produced by an engineer? Do more peer-to-peer communications and dialog via slack decrease the number of bugs in an engineers code?

Probability methods

Traditional probability methods come in crucial for predicting the likelihood of future events happening. They can be used to predict the likelihood of product features or

entire projects being delivered on time. Managers may use these to assist with decisions on whether to hire more developers or to reduce the size of the team to maximize cost and efficiency.

Sentiment Analysis

Sentiment analysis is the automated process that uses AI to identify positive, negative and neutral opinions from text [14]. This new technology will be advantageous to companies wishing to monitor the attitudes of employees towards their work, the workplace and their fellow employees. A Japanese company called AIR is already doing this via its product ‘Vibe’ which works through Slack, a workplace messaging platform. Vibe gives companies “the ability to measure morale in real-time”[15]. It monitors conversations on Slack and tracks the overall mood of employees.

Conclusion

All these analysis methods come in useful for managers when deciding who to hire and fire, who to give a promotion to or who to give more responsibility on the next big project. What’s the probability that a developer is losing satisfaction and motivation in their job given their dip in productivity? Such an observation is crucial for managers working with teams of engineers to ensure maximum output and maximum well being for all employees.

The above methods of analyzing data are useful, but only up until a certain point. For many of them to be effective, they must build upon vast large scale datasets. Therefore a lot of data analysis is limited by the currently available data to draw conclusions from. This is a challenge to be faced by smaller and newer companies with only several projects under their belt.

Ethical Concerns

The collection of employee data is inherently good, and produces many improvements in the world of software engineering and how teams are managed. However, as with all brand new technologies, ethical concerns arise that must be addressed.

The question must be asked: Are customers comfortable with being this closely monitored? From their weekly average code output to the very keystrokes they make, how far is too far when it comes to measuring metrics? The stress of being constantly watched and monitored can have negative effects on productivity and moreover the employee's health. The World Health Organisation notes that workplace stress can arise due to "poor work organization (the way we design jobs and work systems, and the way we manage them)" [16]. This should be strongly considered by companies looking to monitor their software engineers. I personally would maintain that measuring the performance of an employee on a basic level is a good thing, but when it comes to scrutinising their every action, I believe companies have gone too far.

The use of sentiment analysis to monitor the attitudes and emotions of employees could potentially encourage them to fake their emotions and feelings in the workplace. This is profoundly destructive and has many negative effects on the employee and employer. Software engineers will feel under pressure to mask their feelings and mental health. Managers may draw wrong conclusions about how they manage a team when in reality there are many factors externally to the workplace which affect employee emotions.

Many data analysis platforms store and access the data of engineers remotely. In many respects there are no major issues with this provided the information is not personal and it is anonymized. Companies should pay close attention to the data they collect, making sure it is in compliance with GDPR (General Data Protection Regulation). The terms and conditions of analysis platforms should be read and

considered carefully by companies before giving access to employee performance data.

It should also be noted that inaccurate or anomalous measurements can have significant consequences in the workplace. If an engineer has a bad week due to a loss in the family or personal health issues, managers should show compassion and seek to help them rather than scrutinize their ‘off peak’ performance. The work of an engineer should be measured over time, and conclusions should not be made from metrics recorded over short periods.

References

[1]"Software Engineering Overview - Tutorialspoint", *Tutorialspoint.com*, 2019. [Online]. Available: https://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm.

[2]P. Naur and B. Randell, "Software Engineering Concepts and Techniques (Proceeedings of 1968 NATO Conference on Software Engineering)", *Van Nostrand Reinhold*, 1976 [Online]. Available: http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF.

[3]E. Weiss, "Measuring LOC and other basic measurement", 2003. [Online]. Available: https://files.ifi.uzh.ch/rerg/amadeus/teaching/seminars/seminar_ws0203/Seminar_3.pdf.

[4]C. Stevenson, *Software engineering productivity*. London: Chapman & Hall, 1995.

[5]L. Brader, H. Hilliker and A. Wills, *Testing for continuous delivery with Visual Studio 2012*. p. 30.

[6]"ISO/IEC/IEEE 24765:2010", *ISO*, 2019. [Online]. Available: https://www.iso.org/standard/50518.html.

[7]M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia and D. Poshyvanyk, "When and Why Your Code Starts to Smell Bad", *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015.

[8]"Hackystat", *Hackystat*, 2019. [Online]. Available: https://hackystat.github.io/.

[9]"Code Climate vs GitPrime | What are the differences?", *StackShare*, 2019. [Online]. Available: https://stackshare.io/stackups/code-climate-vs-gitprime.

[10]2019. [Online]. Available: https://www.washingtonpost.com/news/business/wp/2016/09/07/this-employee-badge-knows-not-only-where-you-are-but-whether-you-are-talking-to-your-co-workers/.

[11]"What is Computational Intelligence? - IEEE Computational Intelligence Society", *Cis.ieee.org*, 2019. [Online]. Available: https://cis.ieee.org/about/what-is-ci.

[12]"A Beginner's Guide to Neural Networks and Deep Learning", *Skymind*, 2019. [Online]. Available: https://skymind.ai/wiki/neural-network.

[13]"What is Regression Analysis and Why Should I Use It? | SurveyGizmo Blog", *SurveyGizmo*, 2019. [Online]. Available: https://www.surveygizmo.com/resources/blog/regression-analysis/.

[14]"Sentiment Analysis:", *MonkeyLearn*, 2019. [Online]. Available: https://monkeylearn.com/sentiment-analysis/.

[15]J. Cheesman, J. Cheesman and A. Archive, "If You Use Slack, You Can Monitor Company Morale", *TLNT*, 2019. [Online]. Available: https://www.tlnt.com/if-you-use-slack-you-can-monitor-company-morale/.

[16]P. Maulik, "Workplace stress: A neglected aspect of mental health wellbeing", *PubMed Central (PMC)*, 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5819024/.