CSU33012-A Software Engineering

Measuring Software Engineering

Tom Grant 18321531



Essay Outline:

*To deliver a report that considers the ways in which the software engineering process can be measured and assessed in terms of measurable data, an overview of the computational platforms available to perform this work, the algorithmic approaches available, and the ethics concerns surrounding this kind of analytics*

This essay will deliver the report specified in the brief, focusing on four headings. These headings are:

1. How can software engineering be measured?
2. What platforms can be used to measure software engineering?
3. What algorithms can we use?
4. Is measuring software engineering ethical?

# 1) How can Software Engineering be measured?

What is software engineering?

Before we can begin to measure software engineering we must first define it. Below is the definition found on the website Techopedia.

*Software engineering is the process of analyzing user needs and designing, constructing, and testing end-user applications that will satisfy these needs through the use of software programming languages. (15)*

Different methodologies within software development such as agile and plan driven differ in their execution of the functions outlined above, but they do all carry out some form of the above jobs (1) (8) (9).

What are the metrics?

Software engineering has proven since its inception to be difficult to measure and predict. As such it has been referred to as a monster made up of blown budgets and missed schedules (2). There has been much research within academia into its measurement (5). A software metric is a standard of measure of a degree to which a software system or process possesses some property (4) (10). The end product of software design is unlike physical products as much of its performance is tied to how well it can adapt to new demands. The units measure can be broken down into two categories, the quantity of work an engineer or engineers has contributed to a project and the quality of these contributions.

Some metrics which can be used for measurement are:

* Lines of Code (LOC) (12)
* Time
* Function Points (12)

**Lines of Code (LOC):**

The most obvious method of measuring software engineering would be lines of code. In traditional fields a measure of productivity has been how much work is done, whether this is using commission to reward sales reps or paying brick layers by the brick, it ties performance to reward. Lines of code is the output which software engineering produces which most resembles this work. Many influential people within the software engineering field raised issues associated with this measure. It rewarded engineers for stretching out their code to fill more lines rather than finding simple shorter solutions.

*“Measuring programming progress by lines of code is like measuring aircraft building progress by weight.” – Bill Gates*

**Time:**

Time can be broken down into two separate categories, lead time and cycle time (3) (13). Lead time is the time between the products inception and completion. Cycle time is the amount of time a developer actually spends on writing code (3) (14). These two metrics are useful to companies in two ways, lead time allows them to calculate the time at which there costs can begin to be associated with a project, cycle time can provide them with some indication as to the contribution of different engineers. Lead time has always been quite easy to measure, as technology becomes more prevalent in our lives it becomes easier to measure cycle time.

**Function points:**

Function points are a measure of the functionality of a project (14). They count the number of distinct instances of:

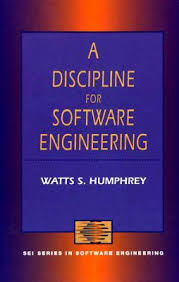
* Input types
* Output types
* Inquiry types
* Internal file types
* External file types

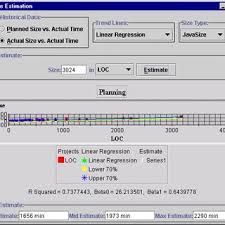
These types are then usually weighted for complexity.

2) What platforms can be used to gather and process data?

## In the world we live in where every move we make on computers can be tracked many companies are trying to analyse our behaviour in order to provide either software engineers themselves or their employers with insight into their work habits. These insights can range from how an employee divides their time between different tasks and how productive an employee is in certain scenarios.

**Personal software process PSP:**

****This is a method of self-assessment with the aim of improving your own software development process (12). It involves carefully planning and recording many aspects of the development process. As it was first conceived in 1986 it was a manual recording process and while you could compare different aspects of the development, there was such a potential for discrepancy that the only reliable comparisons were against your own previous projects.

**Leap Toolkit:**

The Leap Toolkit is an automation of the PSP method and allows for the normalization and of data analysis (12). While it was still up to the developer to enter the data the Leap Toolkit would provide the analysis of this data and provide its own analysis, which was enabled to be more in depth than originally suggested for the PSP analysis as there was possibility of user error.

**Timeular:**

The Timeular Tracker is an 8-sided dice that sits on your desk. Assign an activity to each side and flip to start tracking your time (17). The user can assign each side of the device an activity and turns the dice over when they begin this activity. The Timeular device then analyses how long they have spent on each activity and allows the user to analyse this data using their own methods of data analysis.

**Microsoft Workplace Analytics:**

****Microsoft is one of the largest suppliers of software in the world. 250 Million people use their Office suite of software daily (16). Their software Microsoft Workplace analytics analyses how workers communicate and collaborate in an attempt to help companies and people better understand how they can improve. Analysing employee communication can allow managers to better understand not only employee effectiveness but also their own (16).

**Humanyze:**

Humanyze is a company which analyses how people communicate using a range of mediums in order to collect data on inter employee, employer and client communication. It then analyses this data to provide insight into the efficacy of these communications grouping them (7).

**WayDev:**

WayDev is an analytics platform which collects data about developers automatically about developers. It then analyses this data and provides insight into the various roles of a software engineer providing insight into their use of code repositories as well as providing daily and by project analysis of their efficacy (18).

**Alternate Platforms:**

The above mentioned platforms are by no means a complete list of all the platforms which can be used to analyse software engineering. Plural Sight, Code Climate and flow all analyse Git commits and code quality in order to provide better insight for managers and clients into the software development process.

**What do these platforms measure?**

PSP and Leap Toolkit Measure the time a developer spends on the project as well as their LOC contribution. These two metrics are purely quantitative however and as we look to measure the quality of their contribution we have to look to other platforms. WayDev is a platform that looks more in depth into the coder’s contribution. It analyses the amount they commit how their commits are received and changed as time goes on. This allows you to analyse contributions not only for their size but also their quality. Humanyze and Microsoft Workplace Analytics are two platforms which do not specifically analyse software engineers. They do however analyse many metrics which are applicable to anyone who works in a collaborative environment which includes software engineers. These metrics analyse software engineers for their ability to work within a team which is definitely a qualitative analysis of the standard of the software engineers themselves rather than just their work.

**Limitations:**

The limitations of these platforms is that it can be hard to draw a direct line between some of the metrics that they are measuring and employees’ overall performance. They are however getting a more complete picture of the work practices and life outside of work of software engineers as technology becomes more pervasive. This is encapsulated by how FitBit are also now being used to analyse employees. This more complete picture should lead to more useful and actionable information. Platforms like Microsoft Office Analytics and Humanyze also assume that you can rank and measure communication. We can check how much someone communicates and what the result of these communications are to some extent.

# 3) What algorithms can we use?

**What are algorithms and how do they work?**

The platforms which were outlined above all collect data from different sources and use this data to analyse the software engineers and their work. How the data is used is decided upon by each platform within their development process. Algorithmic approaches to the data use mathematical models to estimate cost as a function of a number of variables which are considered to be major cost factors (12).

Algorithms themselves can be broken down into three distinct groups:

* Linear Models:
  + Cost
* Multiplicative models
  + Cost
* Power Models
  + Cost

In these models the constants a and b are calculated to find those which best suit the project and S is the Code size or LOC (12).

Cost factors are broken down into four groups (12):

* Product factors: Product size, functionality, reusability etc.
* Computer factors: Execution time constraints, storage complaints etc.
* Personnel factors: personnel capability, programming experience, personnel continuity etc.
* Project factors:

**Algorithms:**

**Putnam (Slim)**

The Putnam Model was formed using data from the Norden/Raleigh manpower distribution and their findings on research of completed projects (11).

Putnam’s model is made up of the Software Equation which reads:



In Putnam’s model S represents LOC, td represents the software delivery time and E is the environmental factor which is derived from historical data on the development capability.

Effort is calculated using the equation :

Where D0 is a factor called manpower build-up which ranges from 8-27 based on how much of the software was new or rebuilt (11).

Slim is a platform which uses this model to estimate cost and schedule manpower.

**Constructive Cost Model (Cocomo):**

The Constructive Cost model was developed by Barry W. Boehm (12). The COCOMO is quite simple to understand.

Basic COCOMO use three sets of {a,b} variables. It categorises software by complexity and uses a different set of values for each of these complexities.

Simple: a =2.4, b=1.05

Intermediate: a=3.0, b=1.15

Complex: a=3.6, b=1.20

These are then applied to the Power Model formula and used to provide a rough estimate of cost.

Intermediate and Complex COCOMO use different values for the {a,b} variables

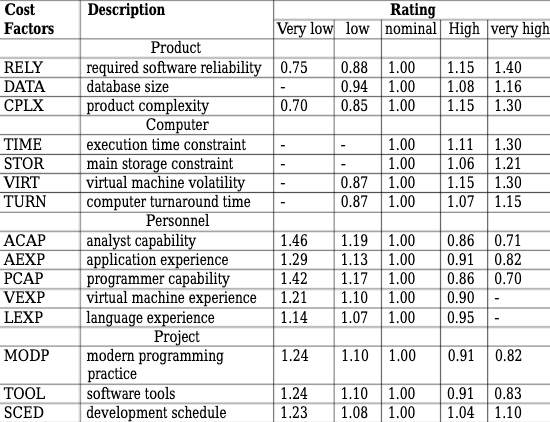
Simple: a =3.2, b=1.05

Intermediate: a=3.0, b=1.15

Complex: a=2.8, b=1.20

There is also a table of 15 other variables which are calculated by analysing the software:

These variables which range from 0.7 to 1.66 are then summed into a variable M. This Mis then used in conjunction with S (the Size of the code in KLOC) and the variables a and b to calculate the cost of the software, again using the Power Formula (12).



**Linear Regression:**

If you use the figures as quoted in the above models there will be a level of inaccuracy due to the disregard of local factors. If you take the log of both sides for the Power function, then you use the least square method to these and make A and B more accurate (12).

**Discrete Models:**

In Discrete Models you use tables of values to calculate cost. Similarly to the table seen in the COCOMO intermediate and complex models (12).

# 4) Is Measuring Software Engineering ethical?

As technology improves and we are able to measure more and more aspects of developers work and lives the question of how ethical this measurement is. In other areas such as medicine, law and economics there are regulatory frameworks in place and ethics can be clearly drawn from these. There is no such framework in the field of performance measurement. In recent years there have been developments in the field of protecting individual’s data (6). All the techniques described above do require data in order to accurately evaluate cost.

The questions then become:

* How should data be collected?
* Who has access to the data when it is collected?
* How should we act on data and insight when it is collected?

How should data be collected:

The GDPR guidelines in the European Union are clear that it is illegal to collect data, store or analyse the data of an individual without their express permission for any purpose other than that which was outlined to them at the time they gave their permission (6).

Personally, I am not completely comfortable with the amount of information which can be collected on me using platforms like Microsoft Workplace Analytics, where not only my work but also the manner in which I perform my job is subject to constant scrutiny.

I do understand however as someone who studies data and works in a family business that when you remove your emotion from the equation that at the end of the day businesses exist to create value for their owners and employees are hired to increase this value.

Who has access to this data once collected?

In the GDPR guidelines it is again stated that an individual’s data can only be accessed and analysed by those given express permission by the individual (6). A way which companies are attempting to sidestep this is by providing their employees with technology as a form of Benefit-In-Kind.

IBM gave many of their workers a FitBit as a present last Christmas. If the company is the owner of these machines, then in effect is it their data to analyse however they see fit. One of the main tenets of the GDPR framework is that there should be transparency around data collection.

When I sit and consider the implications of actions like these, I feel very worried. If companies have access to data regarding your health how much of it would they need to share with you. If as smart watches get more sophisticated, they can predict illnesses, how do we know that companies will not use this as a chance to revoke your health insurance when you need it?

How should we act on this data?

The equation when you look at it in the short term is simple. We need to be as efficient as possible because anything else is unfair on owners or clients. However, when you look to the long term you realise, that there are certain intangibles which exist in every business.

My parents own a school uniform shop which started out as a uniform supplier to the Irish Army in Collin’s Barracks over 100 years ago. One of the women working in the shop has been there since she was taken on to cover for my grandmother going on maternity leave to have my father 54 years ago. She is now 70 years of age and works part-time. The fact that she has been in the shop for such a long time means that when at least one parent a day comes in and reminisces about buying their own uniform from Mary.

If you were to break Mary’s performance down to numbers, you would see that she can’t bring jumpers out quickly, she can’t serve multiple customers at once but what she brings in terms of atmosphere is unmeasurable.

Expanding this beyond just Mary you can see how intangibles cannot be measured and employees can feel unappreciated in a purely data reliant approach.

# Conclusion:

In conclusion, this report has outlined what software engineering is, the metrics which exist to measure it and how these metrics are analysed to measure the engineering. This report has also looked at various platforms which exist to measure software engineering and the ethics around this measurement.

Many aspects of software engineering can be measured and many platforms exist which can provide detailed analysis of areas for improvement and growth. It is important to remember that not all aspects of work are tangible and that we must balance our actions on data and measurements against common sense and compassion. We must remember that many things make up a good software engineer, like empathy and cooperation, and that while not all of these leap out on a stat sheet they are still important.

References:

1. Agile Alliance. 2020. *What Is Agile Software Development?*. [online] Available at: <https://www.agilealliance.org/agile101/> [Last Accessed 27 November 2020].
2. Brooks, 1987. No Silver Bullet Essence and Accidents of Software Engineering. *Computer*, 20(4), pp.10-19. [Last Accessed 27 November 2020].
3. Extreme Uncertainty. 2020. *Cycle Time And Lead Time - Extreme Uncertainty - Practical Agile*. [online] Available at: <https://www.extremeuncertainty.com/cycle-time-ead-time/> [Accessed 27 November 2020].
4. Fenton, N. and Neil, M., 1999. Software metrics: successes, failures and new directions. *Journal of Systems and Software*, 47(2-3), pp.149-157. [Last Accessed 27 November 2020].
5. Fenton, N. and Pfleeger, S., 1991. Software metrics: A rigorous and practical approach. *Software Testing, Verification and Reliability*, 1(3), pp.43-43. [Last Accessed 27 November 2020].
6. *General Data Protection Regulation*. 2018. [Last Accessed 27 November 2020].
7. Humanyze. 2020. *Humanyze Elements Platform*. [online] Available at: <https://www.humanyze.com/elements-2/> [Accessed 27 November 2020].
8. Innolution.com. 2020. *Blog: Scrum And Waterfall: A Comparison Of Plan-Driven And Agile Principles | Innolution*. [online] Available at: <https://innolution.com/blog/plan-driven-vs-agile-principles> [Last Accessed 27 November 2020].
9. Innolution.com. 2020. *Plan-Driven Process Definition | Innolution*. [online] Available at: <https://innolution.com/resources/glossary/plan-driven-process> [Last Accessed 27 November 2020].
10. Kan, S., 1995. Metrics and models in software quality engineering. *Software Testing, Verification and Reliability*, 5(4), pp.273-275. [Last Accessed 27 November 2020]. 9
11. L. H. Putnam, “A general empirical solution to the macro software sizing and estimating problem”, IEEE Trans. Soft. Eng., July 1978, pp. 345-361. [Last Accessed 27 November 2020].
12. Leung, Hareton & Fan, Zhang. 2001. Software Cost Estimation. 10.1142/9789812389701\_0014. [Last Accessed 27 November 2020].
13. P. M. Johnson, "Searching under the Streetlight for Useful Software Analytics," in *IEEE Software*, vol. 30, no. 4, pp. 57-63, July-Aug. 2013, doi: 10.1109/MS.2013.69. [Last Accessed 27 November 2020].
14. TechBeacon. 2020. *9 Metrics That Can Make A Difference To Today’s Software Development Teams*. [online] Available at: <https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams>
15. Techopedia.com. 2020. *What Is Software Engineering? - Definition From Techopedia*. [online] Available at: <https://www.techopedia.com/definition/13296/software-engineering> [Last Accessed 27 November 2020].
16. The Verge. 2020. *Microsoft Teams Jumps 70 Percent To 75 Million Daily Active Users*. [online] Available at: <https://www.theverge.com/2020/4/29/21241972/microsoft-teams-75-million-daily-active-users-stats> [Last Accessed 27 November 2020].
17. Timeular. 2020. *Timeular - Easy And Accurate Time Tracking*. [online] Available at: <https://timeular.com/> [Last Accessed 27 November 2020].
18. Waydev. 2020. *Git Analytics Platform For Engineering Productivity - Waydev*. [online] Available at: <https://waydev.co/> [Accessed 27 November 2020].