Measuring Software Engineering

Nixon Okoli - 17331885

Introduction

This essay aims to explore how the discipline of Software engineer can be measured and utilised in a meaningful way. It will also explore how measuring software engineering can facilitate the development of software, the disciple of software engineering and the management processes pertaining to software engineering.

Measuring Software Engineering is a lot like measuring any profession, thus the measuring process follows the trends of many professions in the 21st century. Technology has taken over the implementation and by extension, the measuring process in a lot of professions and as outlined in the book, “The Future of the Professions” this is a trend that will be more prevalent as the years pass.

Measuring Software engineering is somewhat of an ironically complicated process, it seems like measuring computer engineering would be an intuitive process especially considering the growing trends in technology being used in the measurement process. Software engineering, however, presents some unique barriers to measurement. I will discuss these barriers and possible solutions in my essay.

How can software be measured?

Why it’s vital to measure software engineering

ISO/IEC/IEEE 15939:2017 standard details the measurement process in the ISO/IEC 15288 and ISO/IEC 12207. *“Measurement supports the management and improvement of processes and products. Measurement is a primary tool for managing system and software life cycle activities, assessing the feasibility of project plans, and monitoring the adherence of project activities to those plans. System and software measurement is also a key discipline in evaluating the quality of products and the capability of organizational processes. It is becoming increasingly important in two-party business agreements, where it provides a basis for specification, management, and acceptance criteria.”*[1]

Since quantitative measurement is indispensable in all sciences, computer science practitioners and theorists have been constantly working to apply similar methods to software development. The goal is to obtain objective, reproducible and quantifiable metrics. These metrics may have many valuable applications in planning and budget planning, cost estimation, quality assurance, testing, software debugging, software performance optimization, and optimal human task allocation[1].

Equipped with relevant data, managers can also identify project bottlenecks as early as possible and effectively; reduce risks and eliminate failures. Multiple studies[2] have shown that the longer it takes to find a code error, the higher the cost of fixing the

error.

Measurement systems can also aid in identifying situations where the requirement specifications provided by clients leads to confusion during the software development process. From a 2008 study, it was found that 68% of surveyed companies statistically, were unlikely to be successful due to poor design requirements. More specifically, they likely encountered the following issues:

Have a budget 160% bigger than the original.

End up with 180% longer estimated time than expected.

Provide less than 70% of the expected functionality.[3]

Software Metrics

The process of software engineering leaves behind a vast amount of minable data that can be used to gain insights into the productivity of a software developer or engineer. Software metrics are used against this data to provide measurements for managers and software engineers themselves. A software metric is defined as “a measure of software characteristics which are quantifiable or countable.”[4] Software metrics are often interrelated but also related to the four functions of management: Planning, Organization, Control, or Improvement.

There are a great number of software metrics available for measuring the performance of a software engineer. The software metrics available aren't made equally however and practically only a handful of the available software metrics are used when measuring software programming

Software metrics can be broken down into 4 categories: sized based metrics, function - related metrics, Business/application-specific metrics, Agile metrics.

(Top 10 Productivity Metrics for Software Development | Infopulse, 2020)[3] describes what each metric means and how they differentiate from each other, from his article:

- *“Size-related metrics indicating the size of outcomes from an activity. For instance, the lines of written source code.”*
- *“Function-related metrics represent the amount of useful functionality shipped during a set period of time. Function points and application points are the most commonly used metrics for waterfall software development, while story points are the usual metrics for agile projects.”*
- *“Business/application-specific metrics should tell you how consumers are using your product and whether it’s meeting the market need.”*
- *“Agile metrics should measure different aspects of the development process.”*

A strong measurement practice will employ a selection of software metrics from each

category to increase the productivity and efficiency of the software engineer and also the value and quality of the products in development.

A list of some available software metrics are, but not limited to:

- ABC Software Metric
- Balanced scorecard
- Bugs per line of code
- Code coverage
- Cohesion
- Comment density
- Connascent software components
- Constructive Cost Model
- Coupling
- Cyclomatic complexity (McCabe's complexity)
- Defect density - defects found in a component
- Defect potential - expected number of defects in a particular component
- Defect removal rate
- DSQI (design structure quality index)
- Function Points and Automated Function Points, an Object Management Group standard
- Halstead Complexity
- Instruction path length
- Maintainability index
- Number of lines of code
- Program execution time
- Program load time
- Program size (binary)
- Weighted Micro Function Points
- CISQ automated quality characteristics measures

Size related software metric

Number of lines of code

Source lines of code (SLOC), also known as lines of code (LOC), is a software metric that involves measuring the size of a computer program by counting the amount of lines in the programme's source code.

SLOC is fairly easy to calculate once a decision has been made on what counts as a line of code. There are two main types of SLOC metrics: physical SLOC (LOC) and logical SLOC (LLOC). The specific definitions of these two methods vary, but the most common definition of physical SLOC is the number of lines in the program source

code text (not including comment lines). Logical SLOC attempts to measure the number of executable "statements", but their specific definitions are associated with a specific computer language (for a programming language similar to C, a simple logical SLOC metric is the number of statement terminating semicolons).

The inefficiencies associated with this metric is apparent in its definition. When measuring physical SLOC, irrelevant formatting styles and different coding conventions can significantly change the measurement of SLOC. Also measuring by SLOC does not reward a developer for finding shorter, more economical ways of solving problems. SLOC is also not capable of being used across multiple languages. Logical SLOC may be more appropriate but it is language dependent and more difficult to implement than physical SLOC measures.

Bhatt, Kaushal & Tarey, Vinit & Patel, Pushpraj in their paper, Analysis Of Source Lines Of Code(SLOC) Metric. IJETAE. 2. (2012). Outlines the advantages and disadvantages of using SLOC as a software metric more clearly as well as what text in a programme's source code can be counted. In the paper the disadvantages outweigh the advantages.[5]

Function-related metrics

Function Points and Automated Function Points, an Object Management Group standard

Function-oriented metrics as the name suggests, focuses on how much functionality software offers. But functionality cannot be measured directly. So function-oriented software metrics rely on calculating the function point (FP) — a unit of measurement that quantifies the business functionality provided by the product. Function points are also useful for comparing software projects written in different languages.

Function point analysis (FPA) was the first proposal for a FSM method and it is one of the most accepted FSM methods in the industry. Automated Function Point (AFP) method states the guidelines for automating FPA counting from software source code.[6]

The function points are defined in Measurement Application Application Productivity by IBM's Allan Albrecht in 1979[7]. They determine the functional user requirements of the software and classify each function into one of five types: output, query, input, internal file and External interface. Once the function is determined and classified, its complexity can be assessed and multiple function points can be assigned.

Quesada-López, Christian & Jenkins, Marcelo, conducted an experiment, to estimate the differences between Automated Function point (AFP) and function point analysis (FPA). In their conclusion they found that there were no significant differences

between the methods for functional size estimation, reproducibility, and accuracy.[6]

Since automation seems to not have a significant impact on function point analysis we may see more measurement frameworks utilizing this metric.

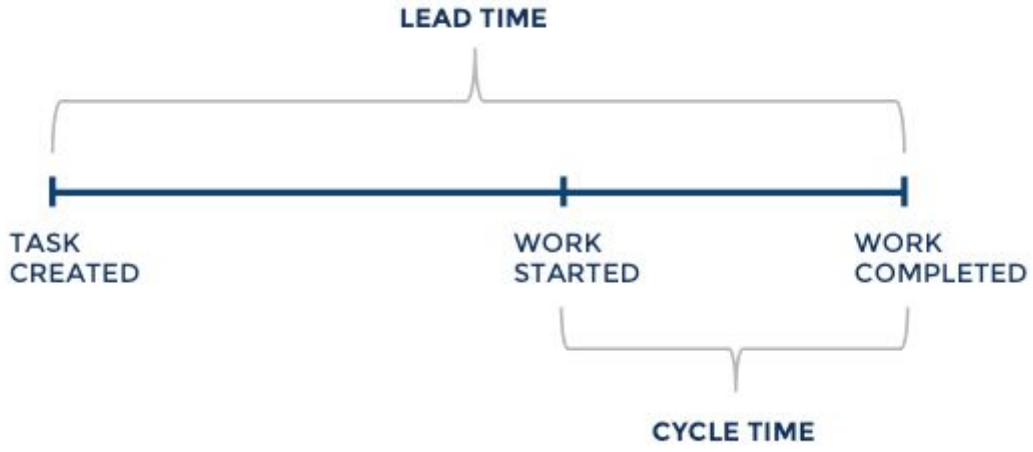
Business/application-specific metrics

Cycle Time

Represents the total time from the start of a project (such as tickets, errors, tasks) to completion.

With this indicator, you can estimate the speed of delivering new features to users. By subdividing total throughput into median time by status or problem type, this is another way to understand how quickly your team is currently completing different tasks. You can identify the exact bottlenecks that affect team performance and set more accurate expectations.

For example, by understanding the average error period, you can convey the correct expectations to the user. By measuring the average functional cycle time, you can manage stakeholder expectations and provide accurate forecasts.



Agile metrics should measure different aspects of the development process.

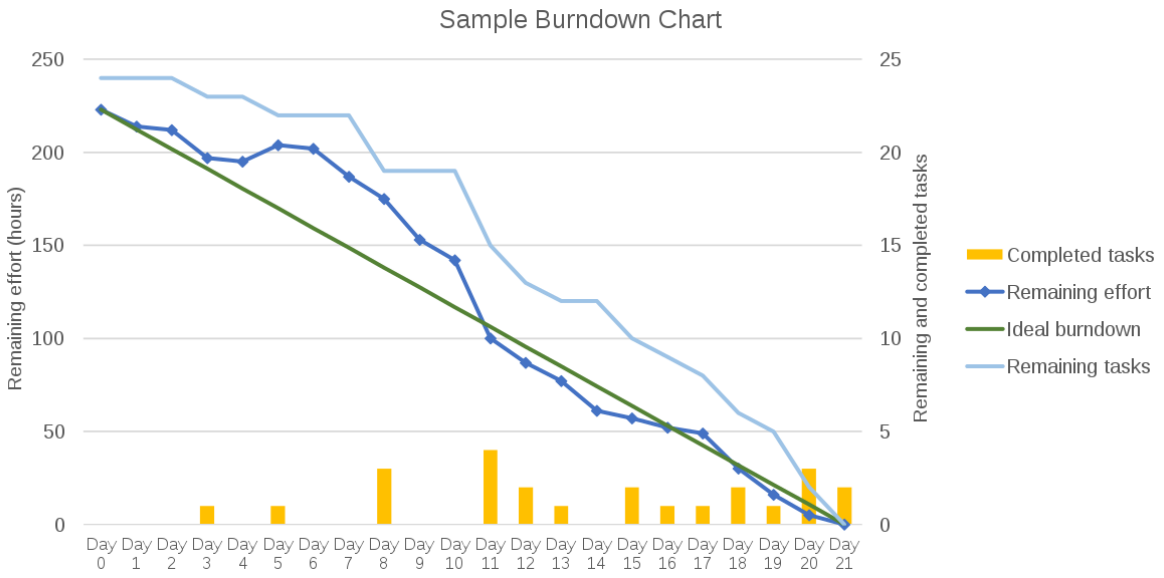
Sprint Burndown

Sprint burndown is one of the key metrics of Agile Scrum. The burnout report conveys the work status of the entire sprint based on the story points. According to the forecast, the team's goal is to deliver all the work consistently.

By tracking this metric, you can gain important insights:

Consistent early sprint completion can indicate a lack of planned work for a sprint.

Conversely, a consistently missed sprint deadline may indicate a gap in your plan, or it may indicate that your team is being asked to deliver too much work. A report should be characterized by a sharp drop in "remaining value", not a sharp drop, because the latter will show that the work is not distributed effectively



What platforms can be used to gather and process data?

Cloud services

Gitprime/Flow[8]

Pluralsight Flow provides workflow analytics for software development teams. By analyzing metadata from git, code reviews, and issue trackers Flow helps engineering teams move faster, increase effectiveness, and debug their development process with data.

Pluralsight flow has a multitude of features that promises to seamlessly integrate into the workflow of a software engineer and boost productivity and efficiency. Their website details what insights their framework can extrapolate from

From their website:

When you understand how your teams are working, you can identify and focus on the work that matters most.

With Flow, you can:

- *See how much time is spent refactoring legacy code vs. new work*
- *Recognize project bottlenecks and remove them*

- *Get concrete data around commit risk, code churn and impact to better guide discussions*

Flow provides a powerful visualization of your team's code review dynamics. Quickly spot unreviewed and old PRs or lengthy discussions in comments. Measure how collaborative a group is and track how these trends change over time.

- *Know if code reviews are productive and positive*
- *Scan for uncertainty and disagreement triggers in code reviews*
- *Observe team dynamics and patterns in the code review review process*

The larger your team, the harder it is to manage knowledge distribution and teamwork dynamics that may be impacting cycle times and code quality. Collaboration in Flow helps you objectively measure these dynamics. You'll see:

- *What percentage of PRs or commits get zero responses*
- *What percentage of the team is involved in feedback*
- *Are senior engineers providing feedback and mentorship*

Get insight into your team's effectiveness by programming language so you can see where they're doing well and where they could use your help. With the proficiency report, you'll see:

- *Commit efficiency by month for each language*
- *Skill IQ data from Pluralsight Skills*
- *Skill development and workflow recommendations for your team*

Example of pluralist's flow visualization

Waydev[9]

Waydev is a platform that works similarly to pluralists Flow. Its main function is to seamlessly integrate into the developers workflow, collect information and data regarding coding effort and abstract useful information for a manager to utilise in their decision making, planning and organising.

So how does waydev measure software development?

Waydev uses several different metrics to provide a “data driven” story for each developer. Waydev looks at the codebase of a developer, the number and content of their commits, the focus of their work through tasks and sprints and an image of how developers work together and their performance as a team.

Frameworks

Hackystat[10]

“Hackystat is an open source framework for collection, analysis, visualization, interpretation, annotation, and dissemination of software development process and product data.”

Hackystat users usually attach software "sensors" to their development tools, which collect "raw" data about development without interference and send them to a web service called Hackystat SensorBase for storage.

Other web services can query the SensorBase repository to form a higher level of abstraction of this raw data, and/or integrate it with other Internet-based communication or coordination mechanisms, and/or generate raw data, abstraction or annotated visualizations.

What algorithms can we use?

Measuring software development really depends on the data that can be gathered and the kind of connections and abstractions that can be made from such data. Thus algorithms that are becoming increasingly prevalent are heavily based on making new connections between data and predicting outcomes and trends amongst developers.

AI and machine learning techniques are at the forefront of new algorithms employed when dealing with large data sets and extrapolating useful patterns and insights that might not be obvious when done through traditional management techniques

For an article[11] at Tech beacon, Anders Wallgreen, CTO at Electric cloud sat down with Stephen Wu, a shareholder at silicon valley Law Group and Peter Gillespie, a

partner at Laner Muchin to discuss the latest ways AI was being used to measure software development.

Wallgren found that AI can help address some of the following, What is contributing to the risk in a particular software release? How much of that risk is contributed by the code itself and how much is due to the developers? Where are teams performing well, and where can they make up ground? What skills does a specific developer excel at, and where does she need more training?

AI driven insights work by performing ML on large amounts of data collected from key parts of the DevOps tool chain to identify patterns hidden in the data that can predict the success of builds, tests, deployments, or overall releases.

Wallgren invites the reader to imagine such a system in his article, "*Envision a platform that would help organizations assign development talent and teams to specific projects by using extremely thorough and accurate data that could indicate how well they would complete that task. Imagine if you could take a project and pick the perfect resource to assign against it every time? That could be a huge value for your business.*"

Is this ethical?

How ethical however is it to collect this information and trust AI to make decisions about human lives?

The ethics behind data collection and usage have been a hotly debated topic in recent years. In this information age, Social Media Today[12] says that in 2020 the average person creates 2.5 quintillion bytes. That's a lot of minable information. The information may be used for any purpose by whoever gets their hands on it. It is well known that in 2016 Cambridge Analytica used the facebook information of up to 87 million people to try and influence the 2016 presidential election.

In Aspects of Data Ethics in a Changing World: Where Are We Now?[13], David J. Hand illustrates the issues and complexities surrounding data privacy and ownership. Before the computer and internet era, data ownership dealt with well defined issues such as copyright issues, patents etc. The ownership of data produced today in this age of the internet is not well defined. Who does the data belong to? Is it the person or company that collects the data that owns it? Or is it the subject of the data that owns it?

Hand notes that "*with automatic data capture and the imminent Internet of Things, it is questionable whether attributing individual ownership of each data item is feasible—we have already commented about how data about one person imply information about other people. Also, further complications arise from derived data, in which the analysis*

and merger of data sets allow the deduction of more information about individuals, which would presumably need to be retained in individuals' data accounts."

With this ambiguity towards data ownership it isn't hard to see how legislators might struggle to set clear well defined principles to guide the gathering of data towards an ethical future. Legislations and regulations have to be balanced for the true benefits of the future of data gathering and measuring to shine. Leading data ethicists Floridi and Taddeo put it like this: *"On the one hand, overlooking ethical issues may prompt negative impact and social rejection …On the other hand, overemphasizing the protection of individual rights in the wrong contexts may lead to regulations that are too rigid, and this in turn can cripple the chances to harness the social value of data science."*

It is quite interesting to note that GDPR does not mention "data ownership". The EU's inception impact assessment of the European Free Flow of Data Initiative in the Digital Single says *"personal data cannot be 'owned' in the EU, but strict rules on access and use by anyone other than the person to whom the data refer are in place," and going on to note that "a gap exists with regard to 'ownership' of non personal data, particularly non personal data that are machine generated.".*

In my opinion, having read the in depth research of many industry professionals and ethics, I believe and trust that despite the possibility for the misuse of data, that the industry will move towards a more ethical approach to measuring data. In the UK the legislation surrounding the gathering and use of storing data include The GDPR Act, The freedom of information Act and the Digital Economy Act. Breaches of these acts can result in fines up to 4% of global Annual turnover. It is my belief that measuring the performance of professionals will be the new normal, thus it is important that we are concerned about the ethics of the inevitable developments and get involved in the discussions pertaining to the ethics of big data.

References

1. ISO/IEC/IEEE 15939:2017(en) Systems and software engineering — Measurement process
2. Bird, J., 2020. The Real Cost Of Change In Software Development - Dzone Agile. [online] dzone.com. Available at: <https://dzone.com/articles/real-cost-change-software> [Accessed 3 December 2020].
3. Infopulse. 2020. Top 10 Productivity Metrics For Software Development | Infopulse. [online] Available at: <https://www.infopulse.com/blog/top-10-software-development-metrics-to-measure-productivity/> [Accessed 3 December 2020].

4. Altvater, A., 2020. What Are Software Metrics? Examples & Best Practices. [online] Stackify. Available at: <https://stackify.com/track-software-metrics/> [Accessed 3 December 2020].
5. Bhatt, Kaushal & Tarey, Vinit & Patel, Pushpraj. (2012). Analysis Of Source Lines Of Code(SLOC) Metric. IJETAE. 2.
6. Quesada-López, Christian & Jenkins, Marcelo. (2015). An evaluation of functional size measurement methods. CIBSE 2015 - XVIII Ibero-American Conference on Software Engineering. 151-165.
7. A. J. Albrecht, "Measuring Application Development Productivity," Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17, IBM Corporation (1979), pp. 83–92.
8. https://www.pluralsight.com/product/flow
9. https://waydev.co/
10. https://hackystat.github.io/
11. Wallgren, A., 2020. Should You Use AI To Make Decisions About Your Software Team? | Techbeacon. [online] TechBeacon. Available at: <https://techbeacon.com/devops/should-you-use-ai-make-decisions-about-your-software-team> [Accessed 3 December 2020].
12. Bulao, J., 2020. How Much Data Is Created Every Day In 2020? [You'll Be Shocked!]. [online] TechJury. Available at: <https://techjury.net/blog/how-much-data-is-created-every-day/#:~:text=If%20you've%20wondered%20how,bytes%20per%20person%2C%20per%20day.> [Accessed 3 December 2020].
13. David J. Hand.Big Data.Sep 2018.176-190.http://doi.org/10.1089/big.2018.0083