Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU33012 Software Engineering
Measuring Software Engineering Report

Merlin Prasad Std# 19333557

November 16,2021

Introduction

In this report I explore how software engineering can be measured, why we would want to analyze this metric as well as the ethical dilemmas this can lead to.

Measurable Data

In order to measure software engineering we need to consider what types of data engineers produce. By analysing these metrics it can be possible to figure out efficiency however the usefulness of many of these metrics is contested.

For the practical piece I examined commits as a metric to measure software engineering and noticed how the number of commits vary exceedingly depending on the workload of the project and the time available.

Below I have outlined some of the more commonly used metrics as well as the benefits and problems that are associated with them.

Agile Process Metrics

Agile process metrics aid agile teams to make plans and decisions . While they don't measure the software itself they aim to help improve the software development process by tracking the progress of a team in creating practical, high quality software that is ready to be delivered.

Lead time
Lead time is a measure of how long it takes to develop an idea and have it as software running in production. Usually you aim to have a shorter lead time to be more responsive to customers . Looking at a team's lead time history can help predict how long a project will take to complete .

Open/close rates
Open/ close rates are a measure of the number of production issues that are reported and closed within a specific time period.

---

https://dev.to/nickhodges/can-developer-productivity-be-measured-1npo
https://blog.pragmaticengineer.com/can-you-measure-developer-productivity/

Cycle time
Cycle time is the amount of time it takes to implement a change to the software system and deliver that change into production. An easy way to game this metric would be to delay when to first commit and then store up commits locally and push them all in one go before the next review. This would hinder the team work process as other developers would be unable to interact with work as it is in process .

Team velocity
Team velocity assesses how many software units a team normally completes in an iteration or sprint. It aids in planning the amount of iterations that would be required. It is however quite futile to compare team velocities as the deliverables of each team would be different .

Production Metrics

These metrics aim to measure how much software is being created and determine the efficiency of the software development teams.

Efficiency
Efficiency is the amount of contributed code that's productive which tends to involve balancing coding output against code longevity. An engineer who is trying a new solution may have lower efficiency rate over an engineer contributing small commits however both are valuable

Mean Time Between Failures (MTBF) /Mean Time to Recover (MTTR)
These technical incident metrics aim to measure how well software recovers and preserves data when a software failure occurs. It can be useful in assessing the stability of a team.

---

https://waydev.co/software-development-metrics/
https://stackify.com/track-software-metrics/
https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams
https://harness.io/blog/developer-productivity/

Code churn

Code churn indicates the amount of change that takes place in a particular area of code for a short period of time. It's useful for gauging code quality. Some level of reworking of a program is expected no matter how skilled the programmer .
Version control systems automatically track code churn using the following formula.

Code churn = Lines Added + Lines Deleted + Lines Modified

It is considered bad if a piece of code undergoes changes too frequently as it's usually a symptom of another issue such as an engineer struggling to write the code required . A high amount of code churn could alert that a project needs attention.

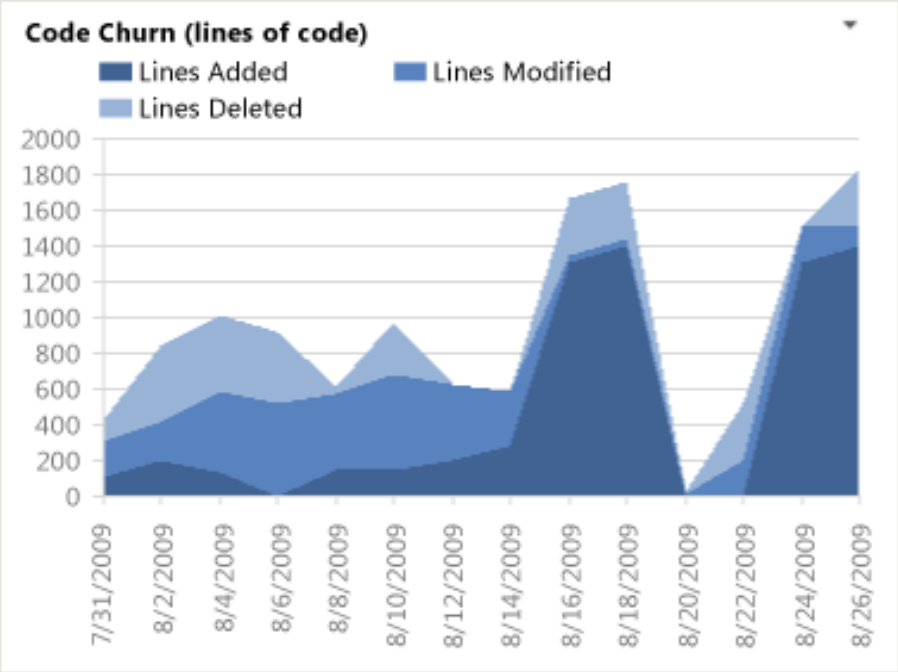

Figure 1 : Shows a visualisation of code churn

---

https://gitential.com/code-churn-the-good-the-bad-and-the-perfect/
https://linearb.io/blog/what-is-code-churn/

Active Days
Active days is the amount of time a programmer contributes code to the project . While spending more time could mean the programmer is putting more effort into work there is also the question on whether spending more time on something should be rewarded. If one programmer is able to write a solution in 2 hours instead of 8 should they be punished if the result of their work is the same ? It can be easily gamed by spending idle hours trying to solve something just to increase your time . Active days are useful to measure to see how much time is wasted on non-engineering tasks such as planning and meetings. It can help plan for more productive meetings and see the cost of interruptions.

Lines of code (LoC)
A common and somewhat naive approach to measure a software engineer's productivity would be counting the number of lines of code they produce. The idea being that if a programmer was solving more things they would produce more lines . However this approach contains many flaws and can be easily gamed. It is possible to solve a problem with 50 lines of code instead of 5 well written lines however this type of measurement would reward the former. This results in people trying to game the system by inserting pointless lines of code simply to be seen as being a good software engineer. This increase in useless code would also be detrimental to the project as these extra lines will require more maintenance. It is quite clearly a poor and unequal way to measure productivity. We wouldn't judge the quality of a painting with how many paint strokes but rather how good the final piece is .

Impact
Impact aims to measure the effect of code changes on the project. Code that affects multiple files would score higher. A few lines of code alterations in multiple files could have more impact then adding many lines to a single file. It aims to measure the efficiency of code changes in a more complex way than LoC

---

https://blog.ndepend.com/alternatives-lines-of-code-loc/
https://www.pluralsight.com/blog/teams/5-developer-metrics-every-software-manager-should-care-about

Development Analysis Frameworks and Platforms

There are numerous frameworks and platforms used in the measurement of software engineering productivity. These tools help gather and perform analytics on the vast quantity of data in order aid in the software development process. I find platforms such as these can be a great tool in helping managers but it also highlights how easy it has become to gather data about software engineering.

Objectives and Key Results (OKR)

Objectives and Key Results (OKR) are a goal setting framework used to communicate what someone wants to accomplish and what milestones need to be met to do this.

One of the issues with trying to use OKR to measure productivity is that it contributes to an increase in technical debt for the company.Technical debt is the cost of additional rework caused by choosing a simple solution at the moment instead of using the more efficient approach that would take longer . Many companies also lack the resources to be able to implement ACK and this puts a lot of pressure on software engineers.

Waydev

Waydev markets themselves as a “development analytics intelligent platform for engineering leaders”. They aim to bring visibility to engineering by moving from a feeling driven leadership to a data driven approach. Waydev track engineering teams output directly from their git repos and generate insightful reports for managers without the input of the engineers. Aggregating and displaying the crucial project and developer metrics in an easy to understand way to help improve productivity.

They want to help companies align their software development process with business initiatives by optimizing the development process.By aligning business initiatives with engineering they can establish unified goals and success metrics. They provide companies with tools such as insights into project costs, help with resource planning and create custom reports.

---

https://www.whatmatters.com/faqs/okr-meaning-definition-example/
https://www.productplan.com/glossary/technical-debt/

Leadership can see how everyone on the team is progressing by gaining a view to the data generated improving visibility. This is done through the analysis of daily standups,one to one meetings and code review workflows. All of this information can help set better expectations by understanding how well teams are working collaboratively and identifying any issues.

Access to objective data helps in managing teams better. Team velocity is improved as the platform aids in the reduction of cycle time and in the understanding of insights in real time . Overall teams perform better with the boost to efficiency , speed up in product development and increased productivity.

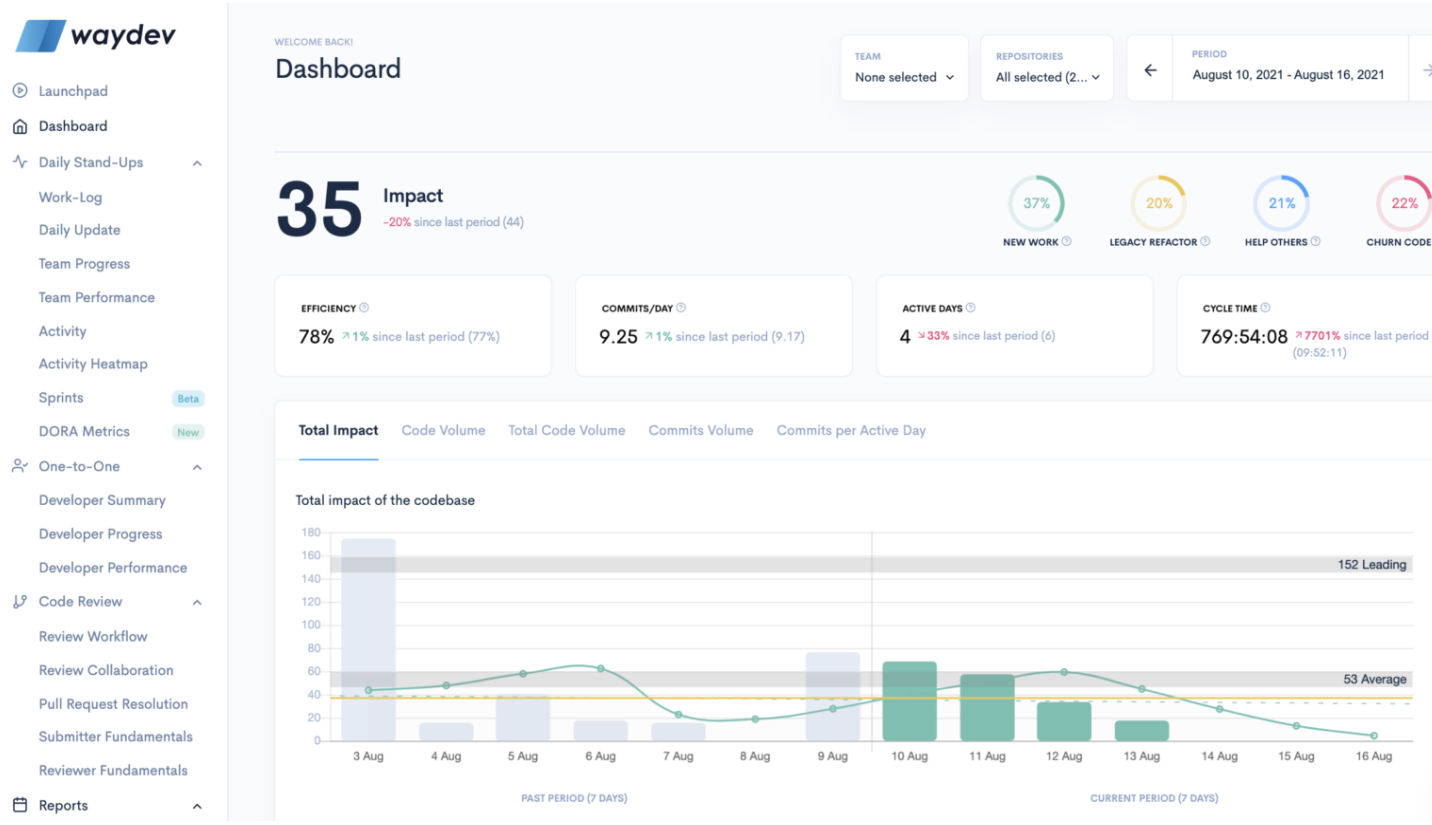

Figure 2 : Shows the waydev dashboard which is used to measure a teams performance over a specific period . Common metrics such as code churn and insights are in an easy to understand format.

---

https://waydev.co/about-us/
https://www.ycombinator.com/companies/waydev
https://waydev.co/gitprime-vs-waydev-vs-code-climate/
https://waydev.co/

Haystack Analytics

Haystack analytics analysis team productivity and efficiency for fast shipping. One of the notable things they examine is the likelihood of burnout for team members by noticing trends such as increased frequency in the completion of pull requests. They gather all their data directly from git to display insights in a simple to understand manner. By providing these metrics they aim to allow for more experimentation and the removal of project blockers which can help reduce the stress of a team. It's interesting to see that companies are trying to measure the health of their workers on top of just their efficiency

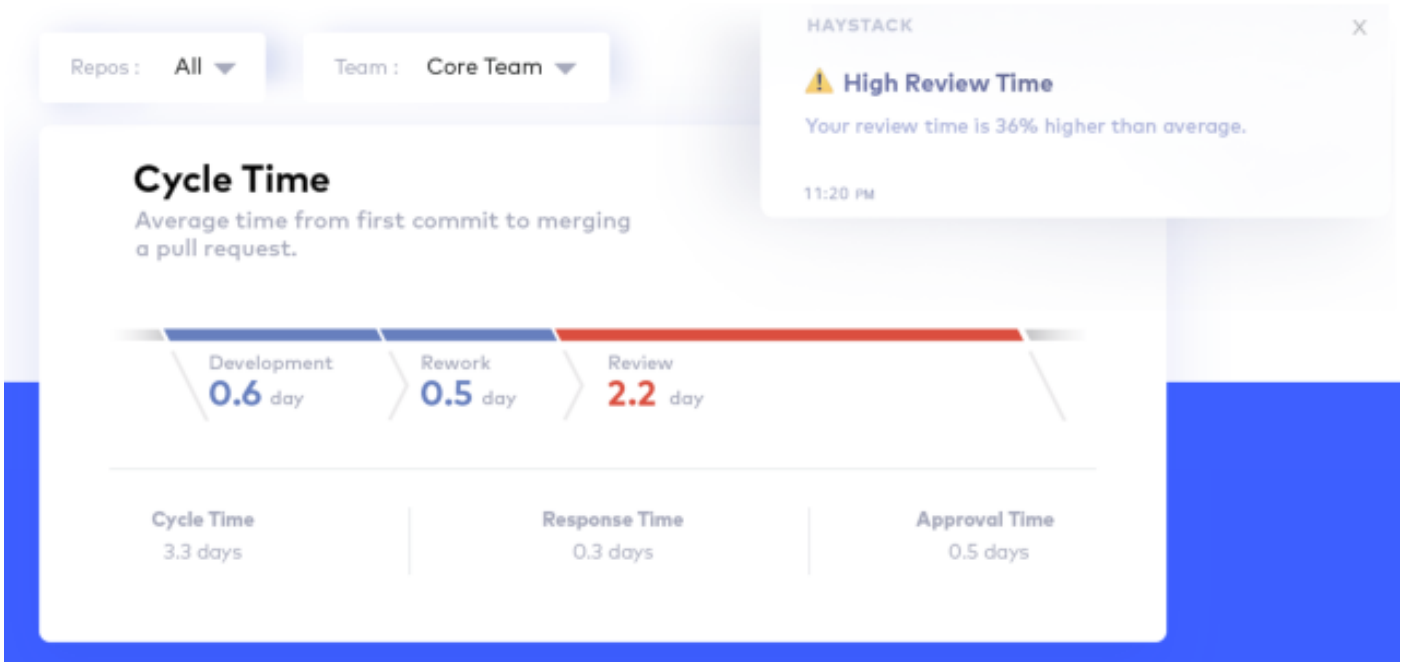

Figure 3 : Shows the cycle time of a team on the haystack user interface. Insights on teams pop up as notifications to quickly alert managers.

---

https://www.usehaystack.io/
https://newsdirect.com/news/83-of-developers-suffer-from-burnout-haystack-analytics-study-finds-978741543

Algorithmic approach

Software engineers employ algorithms to calculate the efficiency and productivity of engineering projects.These algorithms are used to answer questions such as what is good quality code and how complex is the design of a specific program.

Weighted Micro Function Point(WMFP)

This software sizing algorithm examines the code by splitting it into different micro functions. These micro functions all produce a complexity metric which are then used to calculate a single score. It is a relatively new metric created in 2009 as a successor to methods such as cyclomatic complexity and halstead complexity.

The benefits of this algorithm are that it requires less knowledge from the end user and less configuration. As well as this it yields more accurate results than older software sizing algorithms.

The main disadvantage is that due to the complexity of the calculation it needs software to examine the source code and cannot be figured out by hand meaning it can not be used for theoretical educated guesses.

$$\Sigma(WiMi)^{\Pi Dq}$$

M = the source metrics value measured by the WMFP analysis stage
W = the adjusted weight assigned to metric M by the APPW model
N = the number of metric types
i = the current metric type index (iteration)
D = the cost drivers factor supplied by the user input
q = the current cost driver index (iteration)
K = the number of cost drivers

---

https://duecode.io/blog/how-to-check-code-quality/
https://prezi.com/aobfuibuvcfi/weighted-micro-function-points/

Cyclomatic Complexity CYC

Code complexity is calculated by counting the number of linearly independent paths in it. Flow control statements such as while and if can add a complexity of 1 and booleans add 1 depending on the context they are used .To do this a control flow graph is created that represents the program being measured. By examining the edges between the nodes of the graph the complexity of the code is measured using the formula. Code with a lower CYC is generally considered to be better as it's easier to test and prone to fewer errors.

This is one of the most used algorithms for calculating coding quality due to its many advantages. It is easy to and fast to compute . It improves testing by increasing code coverage and highlighting areas that require more testing focus.

Disadvantages with the algorithm are that it is examining the code's control complexity and not the complexity of the data itself. It has a harder time understanding nested conditional statements and can also give incorrect results for simple comparisons and decision structures

$E - N + 2P$
P = the number of connected components E = the number of edges N = the number of nodes

---

https://www.codegrip.tech/productivity/a-simple-understanding-of-code-complexity/
https://www.ibm.com/docs/en/raa/6.1?topic=metrics-cyclomatic-complexity
https://www.geeksforgeeks.org/cyclomatic-complexity/
https://craftofcoding.wordpress.com/2017/06/18/coding-a-small-note-on-cyclomatic-complexity/

```
① start
② if (X) then
③    if (Y) then
④        perform A
         perform B
     else
⑤        perform C
         perform D
⑥    endif
⑦ endif
⑧ end
```

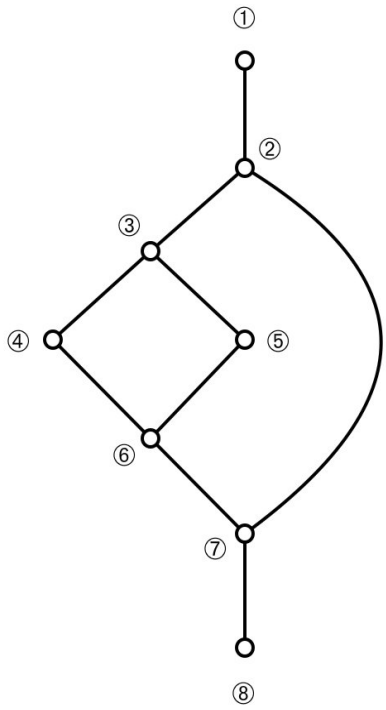

Figure 4: Shows a Flow control graph associated with code snippet on the left .
Using the formula on the above code snippet P = 1, E= 9 and N = 8 .
CYC is 9 - 8 + 2 (1) = 3

Halstead Complexity Measures

Halstead complexity measures aim to measure computation complexity .These measures are a combination of multiple other metrics together in order to compute code complexity. It includes metrics such as programming time, language level, intelligence content, halstead vocabulary and program difficulty .

This algorithm has a range of advantages .It is simple to calculate . It predicts maintenance effort and the rate of errors. It can be used with any coding language and it does not require analysis of the program's design. It also measures the overall quality of the programs.

Its main drawback is that it requires the complete code so it can not be used for predictive estimating.

---

https://www.geeksforgeeks.org/software-engineering-halsteads-software-metrics/
https://www.ibm.com/docs/en/rtr/8.0.0?topic=SSSHUF_8.0.0/com.ibm.rational.testrt.studio.doc/topics/csmhalstead.htm

Ethical analysis

Software engineering is increasingly becoming a field that is more closely monitored with the rise in development analysis platforms. This is resulting in numerous ethical issues that we as software engineers need to be aware of.

One of my biggest concerns regarding the analysis of this data is how we can make sure it is done in an unbiased fashion. There are countless examples of artificial intelligence (AI) being racist such as with facial recognition software google uses being unable to recognise dark skinned people's faces. The systemic discrimination by these algorithims in areas such as health care is leading to peoples deaths. If you did a google search of the words software engineers you will get images of mostly middle aged white men . People claim that AI is neutral but it is clear to me that it mimics the biases of society and the predominantly white males who create it. If we used a machine learning algorithm in the recruitment process for a software company it could very likely lead to minorities being hired less as it would think successful engineers are white able bodied men. Contributions and commits done by white males may be looked on more favourably than those of their brown counterparts. I find this frightening as a brown woman to consider. I would not feel comfortable being completely evaluated by a machine as I do not trust it to make fair and balanced assumptions by itself.I don't think we are at the technological stage where we can take a complete hands off approach and allow our machines to calculate the productivity of an engineer without fixing these overwhelming biases. Machines can not properly measure human behavior and all its complexities. If we wish to create tools that benefit society we need to take into consideration such flaws in technology . This is why I think it's crucial that humans are part of the judgement process when it comes to measuring software engineers as well as in any implementation of AI at the moment .

---

https://www.nytimes.com/2021/03/15/technology/artificial-intelligence-google-bias.html
https://www.technologyreview.com/2020/12/10/1013617/racism-data-science-artificial-in
https://www.nature.com/articles/d41586-019-03228-6

Another key factor to consider is the accuracy of these measurements. Every metric that can be measured can also be gamed . When we measure software engineering are we actually rewarding engineers who are contributing the most or the ones who are clever enough to find the loopholes in the system. Should we be incentivizing people to spend longer hours working if they are able to solve problems quicker? I think the optimum way to ensure fair usage of these measurements is to make sure there are people involved in the process. These technologies can be gamed however a good manager should be able to recognise these patterns and reward work that is actually beneficial to the program. I think the data engineers generate has a lot of potential and we can use these algorithms to give us an easy and quick way to measure the data. However I think we always need humans to be there in the final assessment .

In addition to this there is the issue revolving around privacy. What if the measuring of a software engineer goes beyond the code they contribute and tries to analyze their physical and mental wellbeing. I don't think my employer should be privy to information such as my health as it's too personal. I should be given the final decision on what I disclose with my company . Measuring my happiness using wearable technology frankly sounds intrusive and dystopian . While a happier employee would be more likely to work more effectively, we all have our good and bad days and if we don't wish to share that information we should be allowed that decision. This kind of technology would also negatively affect people with mental health issues. I don't think we need to know all the intimate details of a person in order to see if they are productive . Instead we should respect people's privacy when we use these kinds of measures in measuring them. We should be able to have a level of trust and transparency between workers and companies to have an enjoyable and comfortable working environment.

I believe that if we want to analyze people's efficiency we can use these metrics as a guide but we must always ensure to do so in the least intrusive, fair and balanced manner. At this moment I do not feel AI can be used solely to judge human productivity but I do think that people can use the information gained from these metrics to help in their decision making.

---

https://www.gitclear.com/popular_software_engineering_metrics_and_how_they_are_gamed
https://www.hitachi.com/rev/pdf/2015/r2015_08_116.pdf

Conclusion

In conclusion I think the measuring of software engineering is something that can have a lot of potential benefits. It can boost a company's productivity and reduce their technical debt. Metrics like code churn can highlight issues in the program. Algorithms can help reduce code complexity and improve testing. However I think it is important that we use the right kind of measures when carrying out these sorts of analyses. We cannot fully rely on machine learning to do the calculations . We must instead have people examine the data gathered in order to make judgements and decisions . These platforms that provide metrics are a tool that can be used to help guide us but they cannot be without underlying context of what the work being examined is. By keeping this in mind I think we will also create technology that will help society better as we are aware of the advantages and disadvantages that come from monitoring worker productivity no matter what field.