Introduction

In this report I will explore the world of software engineering including how to measure how productive a software engineer is and what techniques can be used to measure this along with platforms that can be used to do this measuring. I will also talk about computations that can be used to measure productivity of software engineers and ethical concerns with evaluating software engineers as they work.

It is usual for companies to want to see how productive their staff is. This data can help indicate how well the staff is working as a whole and if they are working to the best of their abilities. It can also be beneficial to the workers as they can see how well they are performing over a particular timeframe. In other industries it can be easily calculated how productive workers are. For example, a delivery driver could be evaluated by the number deliveries they make. This isn't the case for software engineering. There are many ways in which a software engineers' performance can be measured.

Measuring Software Engineering

A very simple way to measure the work of a software engineer would be to look at the lines of code written by a certain software engineer. Using version control software's such as Git companies can easily track the number of lines of code that employees write. At first glance this seems like a good idea, the more lines of code you write == the more work that's being done, but this isn't necessarily the case. This can lead to unnecessary lines of code being written to bump up the number of lines of code being written. It also doesn't reward smart/elegant solutions to problems as if someone finds a more efficient solution to a problem with less lines of code, they may keep the less efficient solution with more lines to have more lines written. This can lead to software being inefficient and slower if more steps are being taken to do simple tasks. It also doesn't take into account the language the code is being written in. Some languages might require less lines of code to perform certain tasks compared to other languages.

Another metric that could be used is to track the number of commits that software engineers make to tools like Git. Similarly, to lines of code, measuring the number of commits isn't necessarily an accurate metric. Tracking number of commits would stop people from committing inefficient code and writing lines of code unnecessarily, but instead this can lead to bad practice with people regularly committing small changes to up their commits. If this happens the repository can become crowded with many commits and hard to navigate especially to previous versions if needed. For these reasons I believe using both lines of code and commits to measure the amount of work done can be unhelpful and counterproductive. As you can see below the sometimes having more lines of code isn't necessary as these two segments of code do the same thing but have 3 lines in the difference.

A different way of measuring the engineering activity is by measuring the process and not the output. This focuses on the work being done at the moment rather then the overall goal. By creating regular targets for individuals or groups you can track how many of the targets are being met and in what timeframe. By meeting regularly and discussing what targets have been met and discussing the next targets and how long it should take to complete them you can track the number of tasks completed successfully and which tasks are not being done and who is not completing them.

There are also more extreme ways to measure work done by software engineers like tracking the number/ frequency of keystrokes or mouse activity of a software engineer. This is a bit more extreme as it is always monitoring your activity. I think this isn't a great way of measuring productivity as keystrokes and mouse movement don't exactly correlate to work done. It is also quite invasive as software engineers might feel observed as instead of their commits or lines of code being looked at, they are being constantly watched. It's also not hard to skew the data by spamming random keys of just randomly moving the mouse.

Sometimes the complexity of code produced by a person is taken into account when measuring the productivity of a software engineer. This can be done using a number of factors including number of children of a class, coupling between object classes, or depth of inheritance trees. One of the most common complexity measures is called Cyclomatic Complexity. This looks at different metrics to determine the complexity of a programme.

Measuring Platforms

There are many different platforms that have been made to measure productivity in software engineering. These platforms can measure different metrics or other information depending on what is needed to be measure. Companies choose different platforms depending on what information they are looking to gather. Some of the platforms require the users to manually input information into them to track their progress whereas others may gather information as the users are working on certain files/ code etc. For this reason, there are many different platforms to choose from including Jira, waydev and Pluralsight.

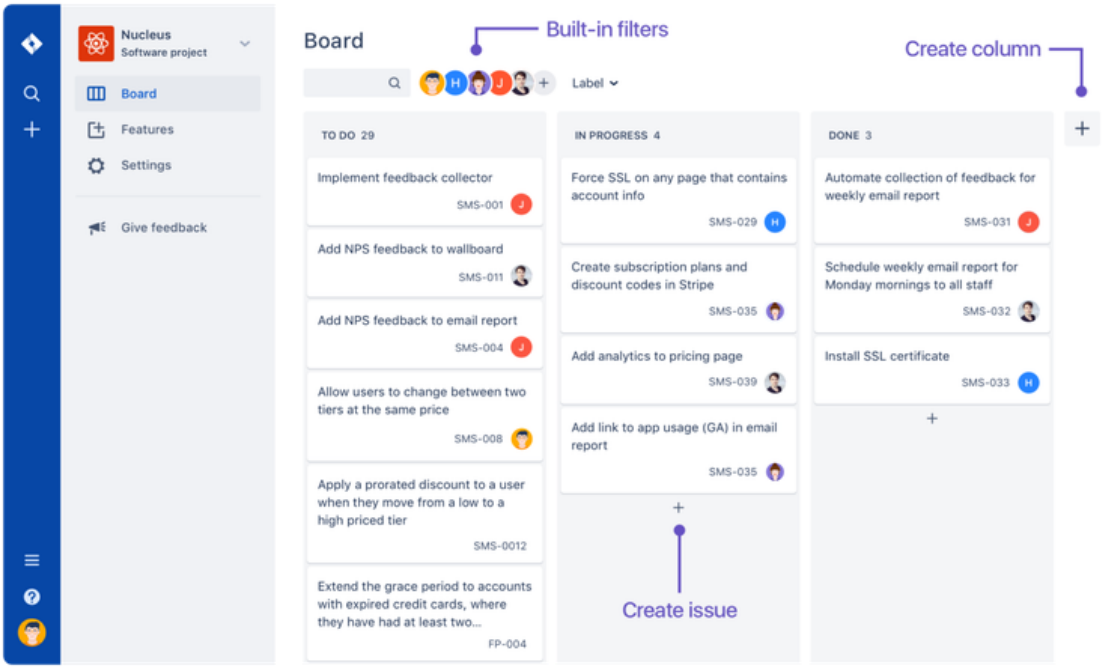
waydev

Waydev is a platform which uses an "agile data-driven method" to collects information about developer teams or developers. It does this by tracking the output from git repos live without any manual output. This allows managers to always have up to date information about all their employees and what they're working on. waydev tracks some of the metrics that I mentioned in the Measuring Software Engineering section such as number of commits and numbers of lines of code written. Although they seemed to not be very reliable ways of

measuring work, waydev takes this information along with other information and it interprets and visualises the data to help set new goals and organise the overview of the project. waydev also takes into account the business side of projects and can be used to connect the engineering team with the business leaders in a company. Although this can be an effective way of measuring the work outputs of a team it can also be very intrusive as the software is constantly monitoring various metrics such as git commits.

Jira

Jira is a platform that measures productivity through agile management. Unlike waydev and other similar tools Jira is more of a manual platform which means that the users have to manually input the information and change it when it changes. Jira uses Kanban software development and uses Kanban boards to show the current tasks and their given statuses. These Kanban boards are made at the start of sprints which can last anywhere from 1-4 weeks. Sprints are short periods of time in which the team must complete a certain amount of work in. At the start of each sprint, they make the Kanban board which outlines who is going to do what bits of work and adds it to the board. All the tasks are initially put into a to do column with in progress and done columns next to it. As developers work through the tasks, they can update the board my moving the tasks into the appropriate columns.



Pluralsight

Pluralsight is the re-branding of Git Prime as it bought Git Prime in 2019. Pluralsight is another example of a more automated platform. Instead of just counting lines of code or git commits Pluralsight measures other metrics. One of the metrics it monitors is the amount of time spent editing old code that may not need any corrections and the time spent on new code. It presents this data and other data including the meaningfulness of the code in code review visualisations which helps to observe the work the team is doing and how the project is being completed. It also checks responses on commits and who is writing feedback on others work in a team. This is a great way of ensuring that everyone in the team is getting involved and are staying informed on the progress of the project. I think personally this is a good platform that has many useful measurements including commit efficiency for different languages, how effective code in different programming languages are and highlighting areas that could be improved on. Having said this some software engineers might find using platforms like this a bit invasive as it is constantly monitoring the work they are doing.

These platforms measure productivity in different ways and all have their own advantages and disadvantages. When it comes to choosing which one to use it is down to the preference of the company and the needs of the company. Some companies might prefer a more relaxed agile management style such as Jira whereas other companies might favour the more intense automation of waydev.

Computational Approaches

There are many different computational methods when it comes to measuring productivity with a ride range of complexity. The most basic computations would be when computing the number of lines of code written or the number of commits done. This is done by adding up the number of commits and lines of code. A more advanced computational approach to measuring the productivity of a software engineer is Cocomo (Constructive Cost Model).

Cocomo is a “procedural cost estimate model” used for projects to estimates the cost for software projects and predicts parameters associated with making the project such as effort, size, time, cost, and quality. The main parameters it measures to show the quality of software products are effort and schedule. Effort is the required labour needed to complete a certain task, which is measured in person-month units. Schedule is the time needed to complete the overall task, which is measured in weeks/ months. This can be a great way for companies to make a timeline of a products development with an end date and knowing roughly how many people will be needed to complete the product.

These characteristics can help determine which system type is most effective for the project an organic, embedded, or semi-detached system.

- Organic: An organic project is one that requires a small team that has a good understanding of the problem at hand which has been previously solved, with the members have similar experience regarding the problem.
- Embedded: An embedded project is one that is the most complex of problems which needs a high level of creativity and experience to complete it. It requires a bigger team to complete with a higher level of experience and creativity among the developers.
- Semi-Detached: A semi-detached project is a project that requirements are between that of an embedded and an organic project.

Sorting teams into these categories can improve the initial layout of a project and allow the right number of resources being used in the right places.

A commonly used software metric for measuring software engineers code is cyclomatic complexity. Cyclomatic Complexity is used to show the complexity of a programme and the level of confidence in a programme. It does this by measuring the number of “linearly independent paths (unique path with loops only counted once)” through a segment of code.

Although this is the most used complexity measurement it still has its downsides. One of the issues it may encounter is the fact that it considers certain statements the same including if, while, for, …. Etc. This can cause issues as a for statement will always be the same code being executed no matter how many times it must execute it whereas an if

statement has different code that is run depending on some variable. Because of this an if statement can have a bigger effect on the complexity, then the for statement.

Artificial intelligence and machine learning is also used to measure productivity from software engineers. Machine learning is a method of data analysis which can learn from inputted data and make decisions with very little input from humans. This is helpful as over time it will become more advanced and better at measuring certain aspects of software engineer's work. This along with other techniques are being used to accurately measure the productivity and work done by software engineers.

Ethical Concerns

When it comes to analysing the performance of software engineers there are a few questions about ethics surrounding this. How far is too far? How much data can they collect before it's an invasion of privacy? There certain types of analysing the performance and work rate of employees I think don't raise any ethical questions such as the use of tools such as Jira or even waydev. Some people might not be comfortable with the use of tools such as waydev as they may feel like they're being micromanaged or that they are being observed but as long as they know that their progress is being recorded and they've agreed to it I don't see any ethical problems with this sort of measuring.

But having said this there are ways of acquiring data which I think can raise some ethical concerns. For example, employers can monitor emails and have CCTV in the workplace as long as they have a legal basis for it. Reading employees emails is an invasion of privacy and personally I think it is morally wrong. Employers can also track other things like keystrokes or mouse movement to monitor the activeness of their employees. I think this is also unethical as it shows a deeper distrust between the employers and employees. I also think that implementing this sort of tracking can lead to unproductivity. If employees are being closely monitored, they may change their behaviour to outsmart the monitoring. This is an example of the observer effect. If the employers start to monitor keystrokes employees will find a way to have constant keystrokes without always doing work. In the same way that writing useless lines of code can skew the data when tracking lines of code written, people will find ways to get around other monitoring.

There are also some legal concerns in some cases of monitoring employees. Employers must ensure that employees have agreed to monitoring as if not it can lead to legal action. Accessing employees' personal materials such as their computer or personal accounts without permission is a criminal offence as it is considered hacking. Personally, I think that ethically speaking it is understandable and ok for employers to monitor heir employees as long as it is done right. Firstly, I think that every candidate for a job should be informed during their interviews that the company monitors its employees and in what ways it does it. This gives the candidate the full information about the job and allowing them to choose whether or not they are comfortable being monitored and how they are being monitored. As long as there is transparency between the company and its employees about monitoring, I don't see why it can't be an effective way to increase productivity and oversee the work being done. Although it could work ethically doesn't necessarily mean it does work. It is unlikely that employees will be happy being closely monitored and could lead to unproductivity, or employees being unsatisfied because of a lack of trust in them because of monitoring.

Conclusion

From all the points that have been made above we can see that there is no perfect answer when it comes to measuring productivity. I personally think the best way of going about it is by putting a bit of trust in the employees themselves. Discuss with them the thought of measuring productivity and what platforms of software they would like to use. Platforms like Jira are great for keeping track of work especially working in groups as everyone can see what stage everyone is at, and everyone is held accountable for their work, and it being done on time. This can lead to increased productivity without invading people's privacy to do it.

References

GeeksforGeeks. 2022. *Software Engineering | COCOMO Model - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/software-engineering-cocomo-model/> [Accessed 3 January 2022].

Library.ndsu.edu. 2022. [online] Available at: <https://library.ndsu.edu/ir/bitstream/handle/10365/25820/Software%20Metrics%20Tool.pdf?sequence=1&isAllowed=y> [Accessed 3 January 2022].

The Man in the Arena. 2022. *What is software complexity and how can you manage it? | The Man in the Arena*. [online] Available at: <https://carlalexander.ca/what-is-software-complexity/> [Accessed 3 January 2022].

Uio.no. 2022. [online] Available at: <https://www.uio.no/studier/emner/matnat/ifi/INF5181/h11/undervisningsmateriale/reading-materials/Lecture-06/morasca-handbook.pdf> [Accessed 3 January 2022].