

Measuring Software Engineering

Introduction

Over the last few decades, software engineering has become one of the most popular fields that people are working in. A survey tells us that there are over 40 million users of Github with more than 190 million repositories. This data is enough to tell us that there are a lot of software engineers out there with a lot of lines of code. In this competitive market of software engineering, measuring a person's or a team's ability to develop software has never been more needed. There are a lot of companies and organisations who are doing this. But there are some questions that must be answered regarding this practice.

How can software engineering be measured?

It has been established that the software engineering has to be measured. But what should the metrics be to measure that?

Length of Code:

The most basic approach to measure software engineering is seeing the length of the code the engineer has committed. This approach may be applicable in a different fields - like measuring the success of a salesman by comparing the sales figures - but, comparing the length of code is not a realistic measure of efficiency and accuracy.

Writing optimized and readable code is a preferred in software engineering so this approach may impact the developer in a negative way.

Hours Worked

Another simplistic measure could be a record of how much time an engineer or a team spent towards developing a specific part of the project. This measure has several disparities like who should decide the number of hours that needs to be spent. Another

problem with this approach is the impact it has on the developer because of the pressure created because of deadlines and limits.

Number of Commits:

This approach can be a good measure to check the accuracy of the code. However, along with the number of commits, one should additionally measure the impact of the new code - check if it creates conflicts, how deep the change reaches the rest of the code. It should also measure the size of the commit as one developer may add many changes in one commit or add small changes many time.

Complexity of the code (McCabe Number) :

A highly complex code can be very difficult to understand for people other than the creator. This could result into a big problem for people working in a team when developing software thus inhibiting the process. Therefore, we can measure a good software code by McCabe Number - which is a measure which quantifies the complexity of a software program. This number is determined by measuring the number of linearly independent paths through the program. A good code would have a low cyclomatic complexity.

Ratio of lines of Code to lines of Test :

A good measurement of software engineering could be the number of tests that the software passes, and if all the edge cases are passed and if the code is reliable. I believe that this approach is the one of the best among all because testing is one of the most important part of software development.

Contributions in Team, Discussions and Forums :

A person's ability to communicate, involvement in discussions and development groups and their participation in various forums could be used as a measure too. In a team, all person's ability to working with their partners and communicating their ideas with each other is really important for a team to perform well. However, in my opinion, this measure would not accurately measure the person's development skill. This could be an additional measure but not the only one.

Technologies and Platforms used to measure it

We have reflected upon the data points that can be used to measure a programmer and the productivity and the efficiency of a team. There are a lot of companies with technologies and tools which use combinations of these metrics to measure software engineering and development productivity.

Some of them are -

Flow by Pluralsight

This product takes data from code commits, tickets, pull requests of the team and visualises workflow and shows where conflicts and problems exist. It sees the behaviour of an individual and the team and analyses that data and suggests sprint plans and also summarises how an individual participates in the team.

Core Features -

- Measures programmer's efficiency by analysing how many times the code written by the individual is rewritten.
- Measures how much new code and work has been done by the programmer by recording the number of commits, number of days spent on the work and other things.
- Determines which team member helps others by measuring the modifications made to other people's code.

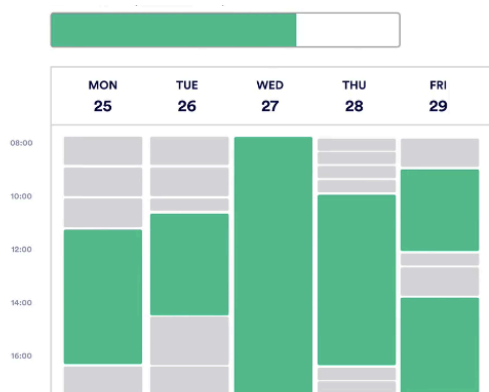


Dewo by Memory


This is a tool developed by Memory which analyses the behaviour of the individual by tracking the person's desktop and web activity and suggests ways to the individual on how to improve their productivity. By tracking app switches, email distractions and meeting schedules, it helps the individual organise their schedule and help them setup for regular intervals of deep work.

Core Features -

- Tracks and visualises time spent of unproductive apps and content switching.
- Gives suggestions to organise meetings providing long windows of deep work to the programmer.
- Manager can track the time spent by the team by integrating Dewo with other platforms.



You normally spend 3h 15m min per day, but this week you got 4 hour and 10 min average per day.

 Slack was your most used app with 16% for a total of 5h 31m.

 Spark was your fourth most app with 9% for a total of 4h 22m.

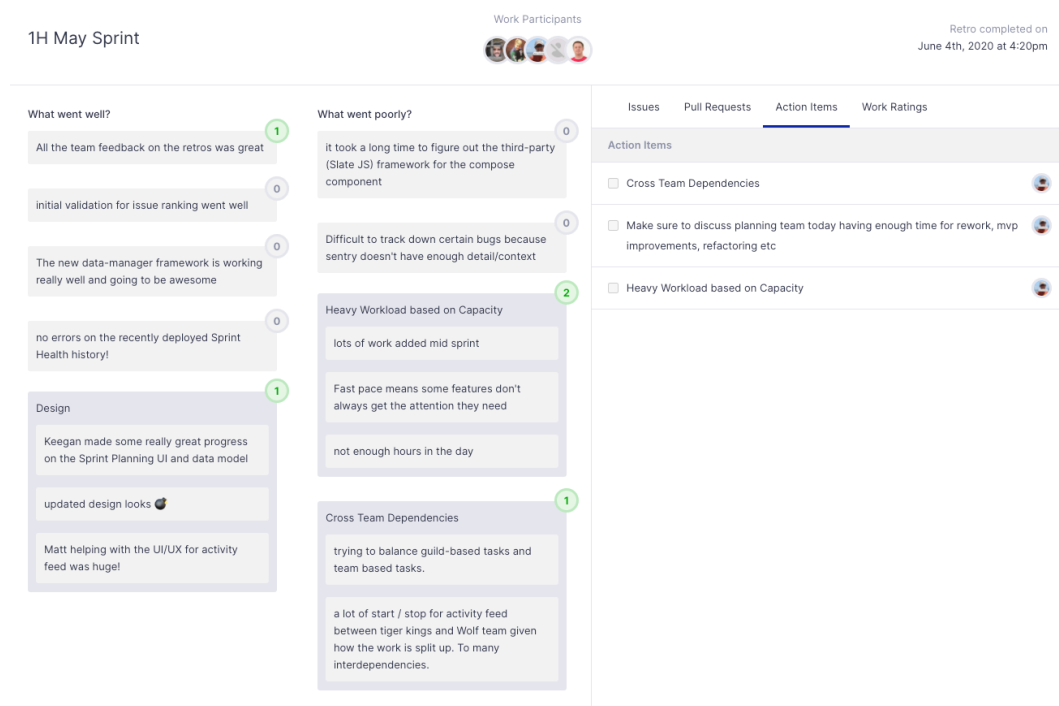
You switch to them 8 out of 10 hours per day for a total of 84 times.

Pinpoint

This solution uses machine learning provides customised recommendations and predicts risk by tracking individual's and team's code ownership, changes per commits and code churn (i.e. code rewritten). Pinpoint is used by many companies because of its simple interface and exceptional features.

Core features -

- Predicts the time it will take to complete an issue.
- Highlights blockers and capacity issues of the team
- Forecasts the likelihood of completion of planned issues in a sprint.
- Automatically assigns and tags issues from backlog to team members.



Pinpoint Snapshot

Waydev

Waydev is tool which analyses Git code and commits by engineers and teams. Using this analysis reviews team collaboration, workflow and develops statistics about programmer.

Core Features -

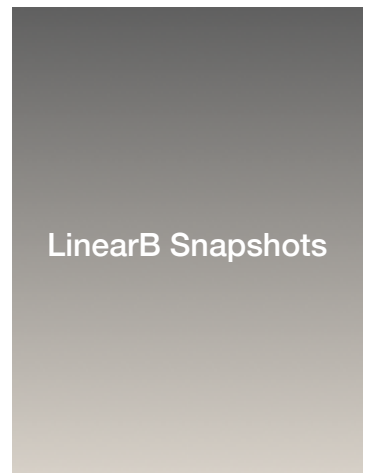
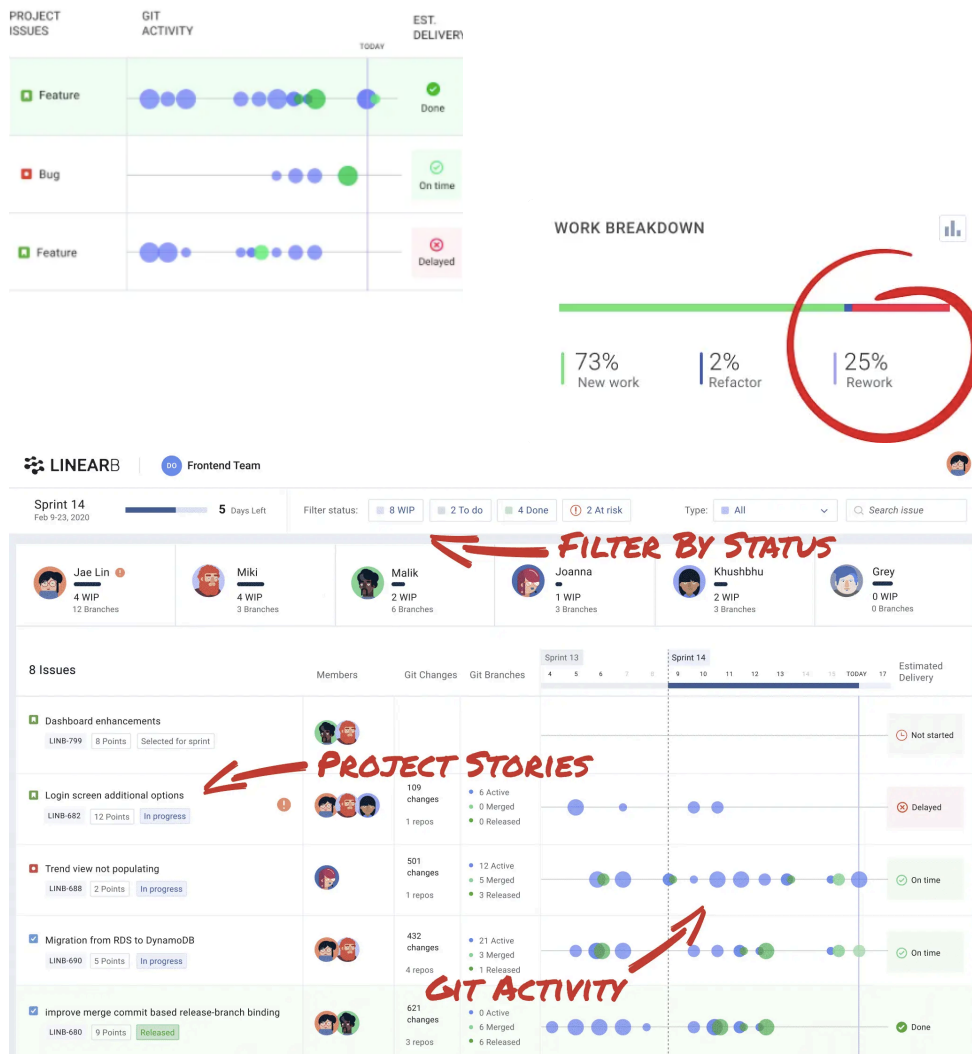
- Develops statical reports based on the programmer's work.
- Reviews bottlenecks and project timelines.
- Visualises Developer Statistics, Comparison and Summary.



LinearB

This tool also uses git activity to highlight important information and find ways to improve team efficiency. It also provides report to the team describing the part of code which looks like high risk code .

- Displays the percentage of new work, refactoring and rework.
- Visualises Changes and Progress in user stories and issues.
- Finds high-risk code, blocks and dependencies.
- Integrates with Slack allowing better team communication.



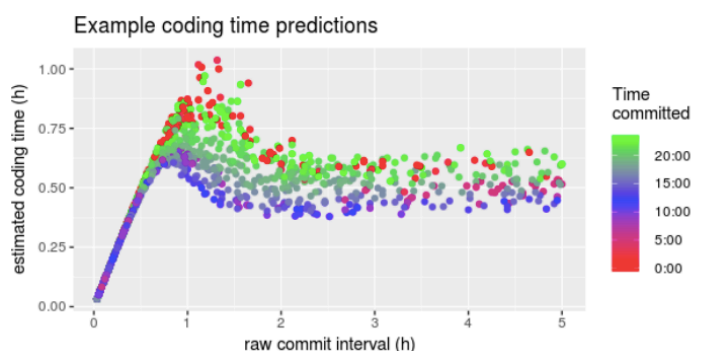
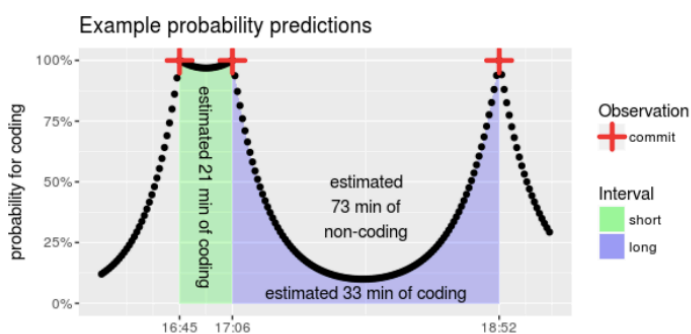
Algorithms & Techniques Used

The simplistic algorithm to measure software engineering is to obtain an average of the combination of metrics mentioned above. This average or a work percentile could be used to compare people and group if the programmer is above or below the average. However, this system could have worse consequences like creating a toxic environment, disturbing team balance and inaccurate judgement.

Therefore, all the modern tools and platforms use a combination of machine learning algorithms to capture data from GitHub code and commits and comparing the data with the user's history and their competitors. There are machine learning techniques which can be used to measure coding output and code quality.

Another model that is used by tools is Neural Hidden Markov Model. It is a statistical model which records data and trains itself which can be used to predict the probability that a programmer is currently coding. This can be used to estimate the expected time a programmer will spend writing a code between two commits.

This is done by recording the time interval between the commits. If the interval is small and regular commits are made that means that the programmer is writing code uninterrupted and. If the interval is long, that means coding interruptions. This is shown in the pictures below -



Ethical and Moral Implications

After discussing the data points, algorithms and tools/technologies that can be used to measure software engineering skill of a programmer, we come to the most important part of the report - Ethical Implications of using this technology.

If we talk in respect of the Data and Privacy laws, it is impossible to summarise how all of these laws are broken by this practice. A lot of tools record the person's web activity and tracks what applications and websites are being used. Many others read and computes what the person writes, speak, communicates along with the time that is spent on doing these things. In the world where we don't want our location to be tracked without our permission, these tools have taken the liberty of recording everything that we do. This is a clear violation of the data and privacy laws.

GDPR outlines rules to protect the users and regulates the process of personal data. Most of the data that is recorded by these tools and frameworks come under the category of personal data. It could be represented as the data owned by the organisation, but the individual is being measured by the tools, not the company.

If we talk about the human side of this technology and by human side I mean the implications above the legality of data capture. Every individual has different traits - we communicate, write, speak, read, behave differently. That is what makes us different. Measuring different people with the same metrics is not an apt way of measuring their skills - No matter how advance the technology is or how many metrics are used. Every team behaves in a different way, every software has different requirements, every developer codes differently and measuring their skills in such a way is unfair to everyone.

I agree that these tools increase the productivity of individuals and teams. Also, it minimises the risk of failure of. It has a lot of advantages which makes measuring like a good way to go.

In my opinion, using these tools at a regulated level is good. If a person or a team is using these tools as just an additional safety measure, then it does not create any harm. But if these tools are used to make decisions and induce any change on the programmer's progress, then these tools have lost their purpose of doing good. All the measuring tools that are available are exceptional but they have to be combined with the correct management systems to ensure that the increased productivity does not come at a much greater cost.

References

- <https://medium.com/scope-ink/measuring-creativity-dfd1168d7c91>
- <https://www.tasktop.com>
- [https://www.chambers.com.au/glossary/mc_cabe_cyclomatic_complexity.php#:~:text=\(Alias%3A%20McCabe%20number\)&text=Some%20can%20avoid%20it.&text=McCabe's%20cyclomatic%20complexity%20is%20a,the%20more%20complex%20the%20code](https://www.chambers.com.au/glossary/mc_cabe_cyclomatic_complexity.php#:~:text=(Alias%3A%20McCabe%20number)&text=Some%20can%20avoid%20it.&text=McCabe's%20cyclomatic%20complexity%20is%20a,the%20more%20complex%20the%20code)
- <https://wiki.c2.com/?ProductionCodeVsUnitTestsRatio>
- <https://memory.ai/timely-blog/productivity-tools-for-developers>
- <https://help.pluralsight.com/help/metrics>
- <https://memory.ai/dewo>
- <https://pinpoint.com/product/for-developers>
- https://www.gitclear.com/pluralsight_gitclear_pinpoint_code_climate_code_development_kpi_metric_alternatives
- <https://semml.com/assets/papers/measuring-software-development.pdf>
- <https://www.productive.io/gdpr/>
- <https://waydev.co/>
- <https://linearb.io/dev-lead/>