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

Measuring Software Engineering Report

Eligijus Skersonas
Student ID: 19335661

Contents

Measuring Software Engineering

Some metrics used for measuring software engineering are:

- Lines of code per person per week
- Test Cases Passed
- Number of commits
- Number of pull Requests
- Gamification

There are many more metrics used to measure software engineering. We will look into some metrics used to measure software engineering.

A common metric used by companies is measuring the number of commits of an individual. This metric involves looking at the frequency of commits made by an individual and the number of commits made. This metric helps an organization determine how productive an engineer is. However this alone is not enough to measure the productivity of an engineer. This metric may be combined with other metrics to help an organization make a more precise decision about an individual's productivity. Some metrics that this can be combined with could be Lines of code per commit, Impact of commit, Test Cases Passed. For example combining the number of commits with the metric test cases passed is effective since an individual could have a large number of commits but pass very few test cases. Combining Impact of commit could help provide an organization with information about how impactful commits are. This helps to determine an engineer's productivity as they may have few commits but those commits may be very impactful.

Another metric that is used by organizations is Gamification. This metric is used to help motivate developers to implement best practices while working on projects. What is Gamification? Gamification is a score based metric. Developers are scored according to their commits, tests, branches and merges. This score based metric can be used to create a leaderboard for developers in a team or organization. This leaderboard will add competitiveness between developers and as a consequence developers will be motivated to increase their gamification score. This will result in developers using best practices for agile development. In the Issues section we will discuss the ethics and legal issues regarding this.
Software Development teams are more likely to be successful using gamification due to the increase in motivation, better practices and competitive spirit. Teams that do not use gamification have a higher probability of lacking motivation which can result in bugs, untested code, failing to meet deadlines, stress, etc.
"gamification is considered a suitable technique for motivational issues in software development"

In the paper "Understanding Gamification Mechanisms for Software Development"
written by Daniel J. Dubois and Giordano Tamburrelli conclude that implementing gamification is an "easy task" however "developing a gamification method and predicting its effect is much more difficult". While gamification itself may be easy to implement it is difficult to develop a method that is effective. There are extra measures to be taken when developing a gamification method, for example making it cheat proof or having the method add little to no stress on developers. These two precautions alone are extremely important to not overlook when developing a gamification method as it could result in very low effectiveness.

If developers are subjected to higher stress due to a poor gamification method then they may burn out over time or they could lose interest in their job due to the higher level of stress. Higher stress levels can also result in conflicts within a team which will impact the effectiveness of the gamification method used immensely. The motivation within a team could potentially decrease, deadlines may be missed and the work environment may become toxic. Hence when coming up with a gamification method it is necessary to not add any additional stress to developers. An example of a method that could add stress include having a leaderboard where developers names and scores are publicly available for the organization members to view. This could result in unnecessary stress for people at the bottom of the leaderboard causing them to overwork themselves even though the work they've done so far is considered to be of good standard.

Also if a gamification method is not cheat proof then motivation may decline amongst developers since their co-workers could potentially be getting higher or equivalent scores by means of cheating while other developers obtain similar scores without having to resort to cheating. This can result in conflict within a team and result in low cooperation between team members and a toxic work environment. Thus when coming up with a gamification method it is important to find any aspect of the method that developers could take advantage of to increase their score. An example of this could be having the number of commits increase your score. A developer could take advantage of this method by committing their code more often, even for the simplest things such as "Added comment to function1", "Added comment to function2". These commits have little to no impact on the project but will increase the comitter's gamification score which may result in the comitter surpassing co-workers who have produced more impactful code.

Gathering Data

There are multiple ways to gather data on developers to measure software engineering. Some methods of collecting data may be:

- Accessing the Github API
- Accessing the BitBucket API
- Taking readings from sensors in office chairs to measure stress levels
- Applying algorithms to CCTV to monitor behaviour

These are some of the examples of how data may be gathered to measure engineering. However there are many more examples of how data may be collected.

Accessing the Github API is one method of collecting data to measure engineering. We can use the Github API to gather publicly available information about any account that is on github. If using python an api you could use is pygithub. Using this API you can get information about a github user's repositories, commits, commit messages, location, email, team projects, etc. You can gather this information and store it in a database for example mongoDB. And then for whatever application the data will be used, the developer can do some data processing to extract information from the database and use that information to measure engineering performance. This measurement can be done using a command line interface or a graphical user interface. A graphical user interface would be a better option as it would be much more appealing and visually simpler to understand the measurement in question.

When performing data processing, depending on the method of measuring engineering it is important to check for outliers in certain data sets and error handling. If one is measuring engineering using a statistical method then checking for outliers is a must because it may cause an imprecise measurement. Error handling is also of great importance, an example of this would be performing data processing on a database containing uncleaned data. This uncleaned data could break algorithms during data processing. It would therefore be important to ensure the data is cleaned before storing it on the database or to clean the data after getting it from the database. For best practices it is best to clean the data before you store it onto the database.

Another method of gathering data for measuring engineering is using sensors in office chairs to measure an individual's stress levels. This can be done by having a heart rate monitor built into the sensor. This could be used for looking after a developer's health. For example if the sensor from the chair of a certain developer is constantly giving high stress readings then the workplace could offer the developer some rest or help with whatever they may be stressing about. This could also look out for potential health problems a developer may have. As a developer sedentary behaviour is an issue and could make them more prone to Atrial fibrillation or higher troponin levels. Atrial fibrillation is the rapid and irregular heartbeat. High troponin levels mean that an individual may potentially have a blockage in a lung artery due to a blood clot or high blood pressure in the lungs. The data gathered from the sensor in the chair could be used to help make developers aware to get up and move around for a bit or even better exercise to look after their health.

Other data that could be gathered is information about employee behaviour. This can be gathered by applying Artificial Intelligence Algorithms to monitor employees and produce data based on employee behaviour. Such data may include if employee x has lunch with employee y or employee x is friends with employee y. This data can be used by managers to help make decisions on creating a team. This information could also be used to detect any conflicts that may exist within the organization as it may reduce productivity, decrease motivation and potentially result in a missed deadline. Hence using data gathered by CCTV a company could try to resolve any conflicts that have been detected by a CCTV camera.

Computations Over Data

There are many methods for performing computation over data. Some examples include:

- Simple Counting
- Linear regression algorithms
- Logistic regression algorithms

These are just some examples, there are many more algorithms and methods of computation over data.

Simple Counting is as the name suggests just counting. This can be very useful on its own. For example it can be used to count the number of repositories a developer has, the number of commits a developer has committed to a project, the number of additions or deletions a developer makes to a project, etc. All these examples use simple counting and can provide a lot of useful data to be used when measuring software engineering. It could be combined with other computations to obtain measurements that are more precise and reflect the work of a developer more accurately.

Another method of computation used is the use of linear regression algorithms. These algorithms can be used to make linear regression models which an organization can use to deduce useful information from. Some information that can be deduced is the average, and the relationship between variables. In a simple linear regression information will be deduced about the relationship between 2 variables. In a multi linear regression information will be deduced about the relationship between multiple variables. A simple example of a linear regression algorithm being used to measure software engineering would be a linear regression model with a Variable for the number of lines of code in a commit and a Variable for the time spent programming. This model could then be used to determine whether there is an increase in the number of lines of code produced when more time is spent programming, if there is a correlation between the number of lines of code produced and the time spent programming etc.

The Logistic regression algorithm is a machine learning algorithm that is used to find the likelihood of success or failure of a given event. This algorithm can be used to predict the likelihood of an event occurring. For example we can use this algorithm to predict whether a developer is going to have tests for the code or not. The algorithm does so by using machine learning methodologies to learn from a dataset the behaviours of a developer. This information can be used to inform a project manager that they should look out for a developer who is predicted to not write tests for their code and to remind them to add some tests. This algorithm is commonly used as it is simple and efficient for binary and linear classification problems.

There are pros and cons when it comes to using different algorithms and methods of computation over data sets. Some pros for using simple counting algorithms is that it is extremely easy to perform and provides simple easy to understand information. Cons for simple counting algorithms would be that you can only get discrete data and cannot make precise measurements when it comes to measuring engineering using solely simple counting algorithms.
For linear regression algorithms the pros would be that we can establish a relationship between variables and check for correlation. However the cons of using a linear regression algorithm is that you may have correlation between variables but there may not necessarily be causality. For example there is a strong correlation between the number of people who died by falling into a swimming pool vs the number of films Nicolas Cage starred in however this obviously doesn't imply causality. Hence further human inspection is required to inspect the relationship between variables for causality. Last but not least the pros of using a logistic regression algorithm are that you can make some useful predictions on the success or failure of a given event. As mentioned previously a project manager could use a logistic regression algorithm to predict whether a developer will commit tested code or not, enabling the manager to prepare in advance to look out for untested code from this particular developer.

Issues

When it comes to measuring software engineering there are a number of ethical and legal issues surrounding it. Some issues regarding ethics would be whether or not an employee would be comfortable with gamification and a leaderboard. This method of measuring engineering has the potential to cause immense stress on some developers who are not committing as often but may have impactful code. This stress could cause a developer to overwork themselves when it is unnecessary to. Another ethical issue would be whether or not an employee would feel comfortable with a sensor in their chair monitoring their heart rate and stress levels or with CCTV gathering data about their behaviours. It could result in psychological discomfort knowing that they are being monitored. The monitoring of individuals using a sensor in their chair may lead to people exercising out of embarrassment rather than solely looking after their health. Using CCTV to gather information about someone's behaviour could result in paranoia among those being monitored and could result in a change in their behaviour.

Some legal issues surrounding the collection of personal data to measure software engineering are GDPR laws. For example, Article 6 states that an organization may not collect personal data unless "You have given your free and informed consent. Your consent cannot be assumed, so silence, pre-ticked boxes or inactivity cannot indicate consent. You must specifically agree to any proposed processing." This means that under no circumstances can personal data be gathered without free and informed consent from the person in question. Thus organizations must abide by these laws and cannot assume consent. A problem that may arise because of this is that an employee may decline to share personal data and hence certain methods of measuring engineering amongst developers may be ineffective if only a portion of the developers consent to the collection of personal data. An example where the measuring of software engineering may become ineffective is when using gamification methodologies to motivate developers to implement best practices for agile software development. Developers who have consented may implement these practices but the developers who did not consent could potentially not be as motivated and not implement best practices resulting in more potential bugs, delayed development and an increase in stress levels.

In some cases measuring software engineering may cross the line. A perfect example is Amazon and their measurement of how long an individual spends in the bathroom. This crosses the line as it causes anxiety amongst workers.
“In the UK, an undercover reporter and a labor survey exposed harrowing work conditions”
A UK undercover reporter found that employees at amazon are afraid to take bathroom breaks in fear of losing their jobs. An employee at amazon stated that they can only be in the bathroom for around a maximum of 6 minutes. Restricting how long a person can spend in the bathroom is extremely unethical in my opinion and individuals should not be monitored when it comes to such private matters.

Another issue that crosses the line in some regard is the use of smart cushions in offices. While they may provide helpful information to monitor employee health, posture and heart rate. It also was able to detect the time an employee leaves their seat. This causes a range of issues such as a manager checking the last time an employee sat in their chair, spelling trouble for employees. This may not be legal because the employee may only have consented to their data to be used for measuring their health and time spent working but may not have consented to use this information to check when they last sat in their chair. There may be other reasons for leaving your chair earlier than expected which a cushion cannot measure. This cushion may also result in an increase in anxiety among workers to not leave their seat. It could also result in unhealthy behaviour if a worker is being monitored for the time spent on the cushion they may be more inclined to sit for longer periods of time promoting sedentary behaviour which contradict the health aspect of the cushion. As mentioned before an increase in sedentary behaviour can lead to a whole range of health problems, from an increase in troponin to an increased risk of atrial fibrillation.

References

Sites

- https://www.getclockwise.com/blog/measure-productivity-development
- https://medium.com/gameful-design/does-gamification-work-in-the-software-development-process-76780e49e545
- https://en.wikipedia.org/wiki/Linear_regression
- https://amplitude.com/https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148#:~:text=Logistic%20Regression%20is%20a%20Machine,on%20the%20concept%20of%20probability.&text=The%20hypothesis%20of%20logistic%20regression,function%20between%200%20and%201%20.blog/causation-correlation
- https://www.theguardian.com/society/2012/oct/30/cctv-increases-peoples-sense-anxiety
- https://www.citizensinformation.ie/en/government_in_ireland/data_protection/overview_of_general_data_protection_regulation.html
- https://www.theverge.com/2018/4/16/17243026/amazon-warehouse-jobs-worker-conditions-bathroom-breaks
- https://www.nytimes.com/2021/01/12/world/asia/china-office-cushion-surveillance.html

Studies

- https://dl.acm.org/doi/abs/10.1145/2591062.2591148
- https://www.sciencedirect.com/science/article/pii/S0735109714071745
- https://www.nature.com/articles/s41598-019-49686-w
- https://www.researchgate.net/publication/269715361_Ever-present_threats_from_information_technology_the_Cyber-Paranoia_and_Fear_Scale