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

CSU3301 Software Engineering

Measuring Software Engineering

Saumya
Bahuguna
21344349
11th January
2022

Contents

1 Introduction

"If you can't measure it, you can't improve it."

-Peter Drucker

In this report I will discuss how software engineering process can be measured and assessed in term of measurable data that is obtained by different computational platforms, I will also discuss the platforms that provide the data and what type of data is provided and useful, and will be proceeding with discussing what type of algorithms and computations can be performed on this data in order to profile details of a software engineer, and in the last part I will be presenting my views on the ethical obligations behind the processing of the personal data.

Before diving deep into the main sections of my report, I would like to first put some light on why measuring software engineering is required.

Measuring a Software Engineering process of a developer team is a very important part of managing a team and helps working under budget by reducing costs and allows team managers to know about their team's performance. It helps to identify and spread more productive development process across the organization, it also helps to identify the high performers and motivate or reward them, it also helps in identifying low performers and the field they are low in which in turn helps the managers to provide the required help if necessary.

2 Measurable Data

Code Churn:

Code that is rewritten or deleted shortly after being written is known as Code churn it is a very common part of the development process. An analogy in layman terms to understand the importance of code churn is: If a person writes 4 postcards and only sends one of them the time taken to send a postcard will also account the time taken to write the discarded postcards not to mention the wastage of resources. When applied to a big scale churn effects teams, an engineering with high code churn rate will be spending more time on

writing a deleted code which results in inefficiency. A high code churn rate is not bad as working on a feature, testing it and refining it by reworking are part of the development process. It becomes bad when the code churn elevates as the deadline approaches, as compared to a code where code churn is low a code with high code churn will have a possibility of having more defects and problems as the code got edited several times or in a large quantity.
Code churn also shows the lack of skills of a programmer or a Poor communication with the client on what is required, both of which can be bad for a company and a team. Figure 1 shows total code pushed by three developers in a duration of a week and as is clearly shown in the bar graph, Jason has more line of codes than other developers but that doesn't mean that the code was efficient, a metric based on this graph is a process named as Line of code, where an engineer is measured with the amount of line of code they wrote.

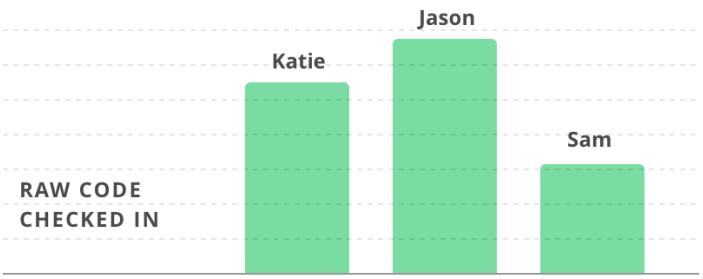
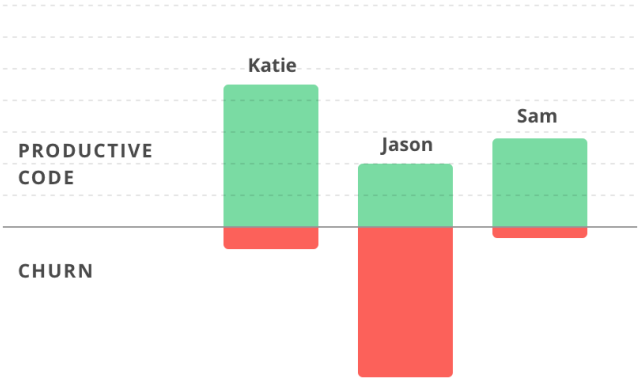


Figure 1: www.pluralsight.com

Unlike Line of code, churn measures the code that was efficient or was used, as shown in Figure 2, Jason deleted a lot of code and was less efficient compared to others. To Conclude code churn can be used as a very beneficial metrics in measuring a team and to let the managers or team leaders to make their team more efficient and productive



As mentioned earlier Line of Code is line of text in a code that is not a comment or a blank, Line of Code is one of the early metrics to measure software engineering, it has faced several criticisms as it measures an engineer based on the lines they wrote, which is not an efficient way to measure programming.

Testing-

One of the most important processes of software development is testing, as testing a codebase will make it less likely for it to fail and increases the quality of code provided, testing is not a metric, but a codebase can be checked for testing i.e. the range of code that the testing covers and the feature it testes as a large untested or unexecuted not only increase the codebases volume but also can be a potential reason of future problems or making the code less readable. Although testing in a code base decreases the possibility of errors, but the test themselves must be all-inclusive and should keep in account the edge cases and different types of inputs. To conclude an engineer can be measured with the tests they provide for the code base and can portray the power and importance of the code base.

Code Review Turnaround Time

Reviewing a code is a one of the important steps in software development as it helps in cleaning lots of Bugs, un-usable line of codes etc. but sometimes code review turn around time is huge as effects the efficiency of the process, it can be solved by turnaround time being specified by the reviewer as , no one else more than the reviewer knows the time it will take them to review, hence measuring the turn around time of the review based on the specified time at the beginning as Code review is a very important and if hurried can result in many potential problems or more inefficiency.

Conclusion

To conclude this section of the report there are many metrics that can be used to measure software engineering process and it's not about choosing one process as many metrics can be clubbed together and can be measured based on the importance of the metric. I have just listed the metrics that are useful to measure codebase of a engineer, there are metrics to check the health of the employee or social behavior of the employee as similar to measuring the codebase measuring the health of the employee is also important.

3 Computational Platforms

The number of Computational platforms available to measure software engineering is high I will be discussing some of them in this section:

Waydev

It uses data from source control platforms i.e. git, to analyze the data about the employees of a company for efficiency and uses different metrics to measure software engineering. The advanced and comprehensive tools it provides, helps the management to evaluate process of developers. The methods are divided into three major categories:
By visualizing the methods, habits and contributions of every team member on a daily basis, this method makes it easy to focus differently on different employees as the data to measure is less on a daily basis, they use this data to set achievable goals and to make better decisions. The next method they use is one to one meeting as to have a sufficient amount of data and metrics for every member means that conducting one to one meetings can accurately reflect and address a members performance. The other method that they extensively focus on is their code review workflow where they provide an aerial perspective of the merging and pull request activity providing management with a full idea of the process and optimize the developer collaboration.
Waydev uses and improves tools like Jira to generate react to understand insights and reports for managers. It uses different metrics and provides the measure in a form of reports as a summary for managerial use.
Some of the metrics used by Waydev are:
- Impact – a way to measure the weight and impact of the code changes happening, much more advanced than methods like simple line of code

analysis.
- Throughput and productive throughput – referring to the total amount of code, be it new, churned, or refactored. Productive refers to the proportion of code committed without churn.
- Efficiency – what percentage of a developer's code is productive. The higher the efficiency the longer the code has been providing business value.

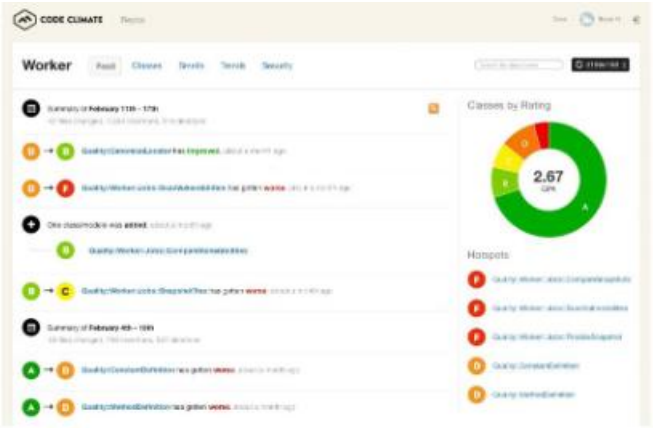
Important to note that a high churn rate will decrease this metric.
- Technical debt – the amount of code refactoring done by the person

Hackystat

Is an open-source framework which provides features like collection, analysis, visualization, interpretation, annotation, and circulation of software development process and product data.
Users typically attach software sensors to their development tool which collects and sends the raw data to Hackystat Sensor Base for Storage, the SensorBase repository can then be used by other webservices to perform higher level abstractions of this raw data or it can be attached with other internet-based communication or coordination mechanisms, or to generate visualization of the raw data. According to them the long-term goal they are trying to achieve is to facilitate "collective intelligence" in the field of software engineering by providing collection, annotation, and diffusion of information and its subsequent analysis and abstraction into useful insight and knowledge. It can be attached with source control tools like git so that a visual insight of codebases is provided to the users. There are a few concerns that come by using Hackystat as it uses automated data recording which can be unsettling for users as the amount of data collected is uncertain.

Code Climate

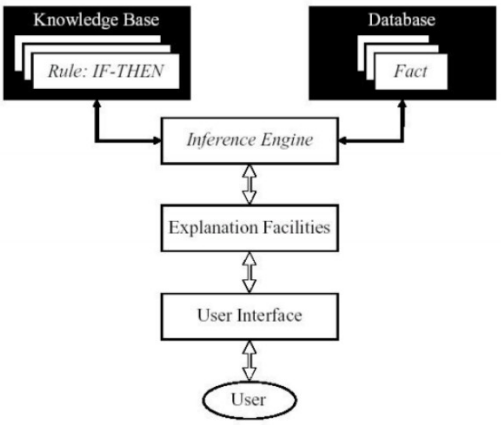
Is one of the relatively modern computational platforms available.
It can be combined with git and is used to determine the quality of a code. The code is analyzed for complexity, duplication, and other changes in quality and technical difficulty after each push, it comments on the users pull requests by sending automates code reviews, which allows for assessment and potential improvement. Automating the platform reduces the chance of human error. Code Climate provides an objective analysis of the quality of your code and gives you the information and tools you need to fix it. (Code Climate, 2019)



4 Algorithmic Approaches

Expert Systems

They use Ai to simulate the behavior and decision making of a human or an organization with expert knowledge.
It consists of a knowledge base and an inference engine which contains organize collection of facts about the system's domain. The inference engine check and matches the facts in the knowledge base to provide and answer the typical tasks in an expert system are to involve classification, diagnosis, monitoring, design, scheduling, and planning for specialized endeavors, it is helpful for measuring a code and automating reviews. The facts in a knowledge base of a Expert system are provided by human experts through different observations and interviews



The figure shows a general structure of an Expert system which consists of these components:
User interface: - it takes a user's query in a readable form and passes it through Explanation facilities to a inference engine.
Inference engine: - it is the decision maker of the expert system. It contains rules to solve problems it refers to knowledge base.
Knowledge base: - it is a database with facts and if- then clauses. With all the possibilities.
It has many strengths compared to a human expert as it is easy to document, it can be easily transferred, it lacks human error, is consistent and predictable and at last, it is cost effective. It can be very useful in measuring software engineering by providing the knowledge base with corrects facts about the metrics and data. There are also some weaknesses of a expert system over a human or conventional system i.e. it can not make creative decisions in some uncommon situations, a simple mirror in a knowledge base can result in big differences in decisions, it is too expensive to maintain it.

Conclusion
An expert system uses Ai and different components and its own knowledge base which is developed by large data and human experts.
it can be very beneficial as it has many strengths with less weaknesses
But as it is costly to maintain and can be useless in uncommon situations it can be imbedded into software but is inefficient to be used solely for a company

Deep Code

The application of machine learning and artificial intelligence into the code review process is becoming very common. Related to that Deep code is a product by snyk company, according to them it is a first-real semantic code analysis tools, which scans code fast enabling real-time workflows within the development process. it draws upon all its users to combine a general knowledge base that teaches itself to recognize common coding practices errors and suggests fixes based on its knowledge base. It consists of tools which can be integrated with an IDE or a source control system, which allows it to automatically review of code or pull requests, even though this is not a metric or a measurement tool, but it can result in significant increase of code review times and accuracy.

5 Ethical Concerns

In this section I will presenting my views on the Ethical issues and obligation around Measuring Software Engineering. After researching and reading through many reports and methods that are being used to or are proposed to use for measuring software engineering, I do not have a clear sided opinion on the topic and will be pointwise stating it for different metrics. Measuring software engineering is a very difficult task as there is no specific method or a specific value to measure. There are numerous methods and processes in Software development all offering important values, Therefore, there are different metrics which not only measure the codebase but also records the movement and daily work life of a developer to make reports, but in my opinion a software developer knows how important that data is to analyze and manage a team's work or to deliver a product, though there are limitations on what data can be recorded and stored most of the data if is stored keeping in mind all the privacy concerns, can be very beneficial for the company as well as the developer, but no metric or system which breaches the personal space of a developer should be used without their consent. On the other hand, the data that does not breach any privacy i.e., churn, testing, and other metrics that are performed on the codebase data can be really helpful for a company and developers, but should neither be used to penalize nor to result in developers losing their jobs as mentioned earlier software engineering has various kinds of process and methods which can be subjective, and a simple metric cannot decide how efficient or inefficient a developer is by a pattern (exceptions exist).

6 Conclusions

To conclude, as mentioned several times in the report Software Engineering is a very large process and has vast number of methods and Metrics. Therefore, it is beneficial to not follow a single metric but to use different metrics and weight those metrics based on the product or framework used. Different companies have also used peer to peer review to analyze their developer according to me that is one of the efficient methods to analyze a developer, rather than tracking them.

References

- https://www.pluralsight.com/blog/tutorials/code-churn/
- https://waydev.co/waydev-metrics/
- https://code.google.com/archive/p/hackystat/
- Code Climate, 2019. [Online]
- Available at: https://codeclimate.com/quality

- Sociometrics in Software Engineering: Measuring Technical Skills of Software Engineer through GitHub
  Rohan Bagwe

- https://snyk.io/blog/accelerating-developer-first-vision-with-deepcode/
- https://www.techtarget.com/searchenterpriseai/definition/expert-system
- https://www.guru99.com/expert-systems-with-applications.html