OLIVER FLYNN - 19336592

MEASURING SOFTWARE ENGINEERING REPORT

CONTENTS

Intro ..... Pg. 3

Section 1: methods of measurement ..... Pg. 4

Section 2: Platforms for measuring data ..... Pg. 6

Section 3: Computational methods ..... Pg. 8

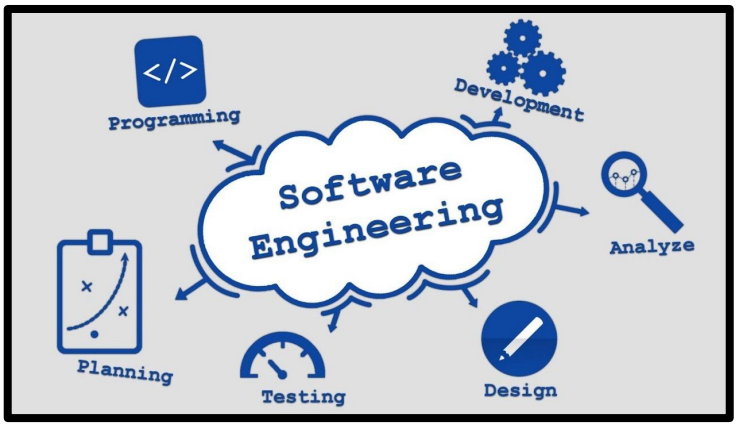
Section 4: Ethical concerns ..... Pg. 8

Conclusion ..... Pg. 10

INTRODUCTION

Software Engineering is a practice that has been around since the second half of the 20th century and has quickly become an industry sector which has created endless career opportunities and helped create and develop software systems, we as a population use daily. A "software engineer" is the title given to the one who goes through the steps of building software from scratch to end up with a final working product, or anyone who practices any parts of the software building process. There are several standard industry practices and tools used to assist a software engineer in the process of creating software, including agile development, unit testing and version control software used to help teams work together on developing an application or piece of software.

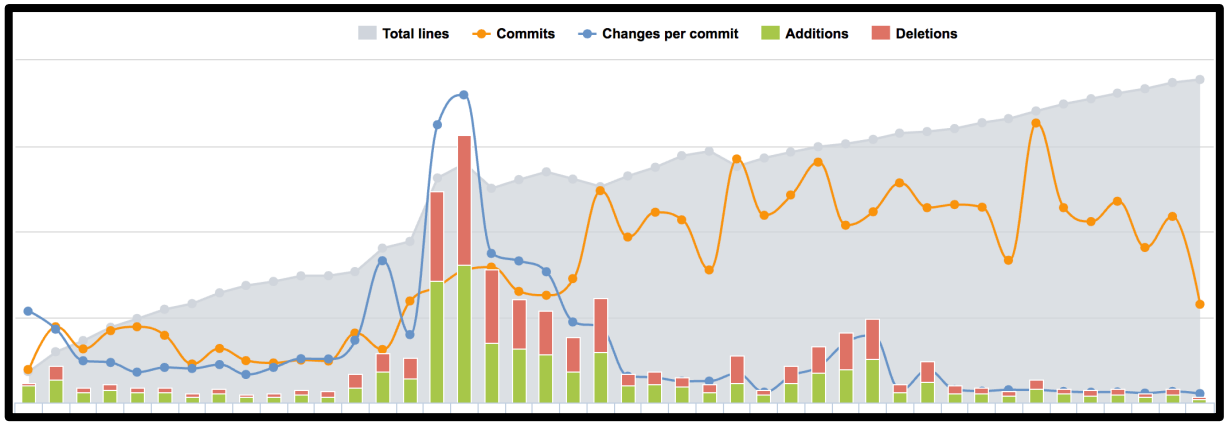
This report intends to explore and enquire about techniques that could be used to track and analyze habits and general data surrounding the routine of a software engineer. The structure of this report is split into four sections as well as an intro and conclusion, section one will explore techniques that one can use to measure engineering data. Section two should explore the platforms on which one can gather and perform calculations on the data, whereas section three should focus on the algorithms and calculations that can be performed on the data. The final section, section four, should investigate the ethics of measuring software engineering and report weather it is ethically and morally right to conduct collection of data on software engineers or not.



SECTION 1 – METHODS OF MEASUREMENT

With today's technologies and practices used in the engineering process it is very easy to collect data on varies aspects of the journey of a software engineer. In this section I will investigate various ways one could quantitively measure different aspects of an engineer's journey in the field. I will first discuss measuring general activity in the industry, like time spent working, software packages created things like that. Moving on I will cover the interaction a software engineer has with peers, and finally I will go over the work patterns of an engineer.

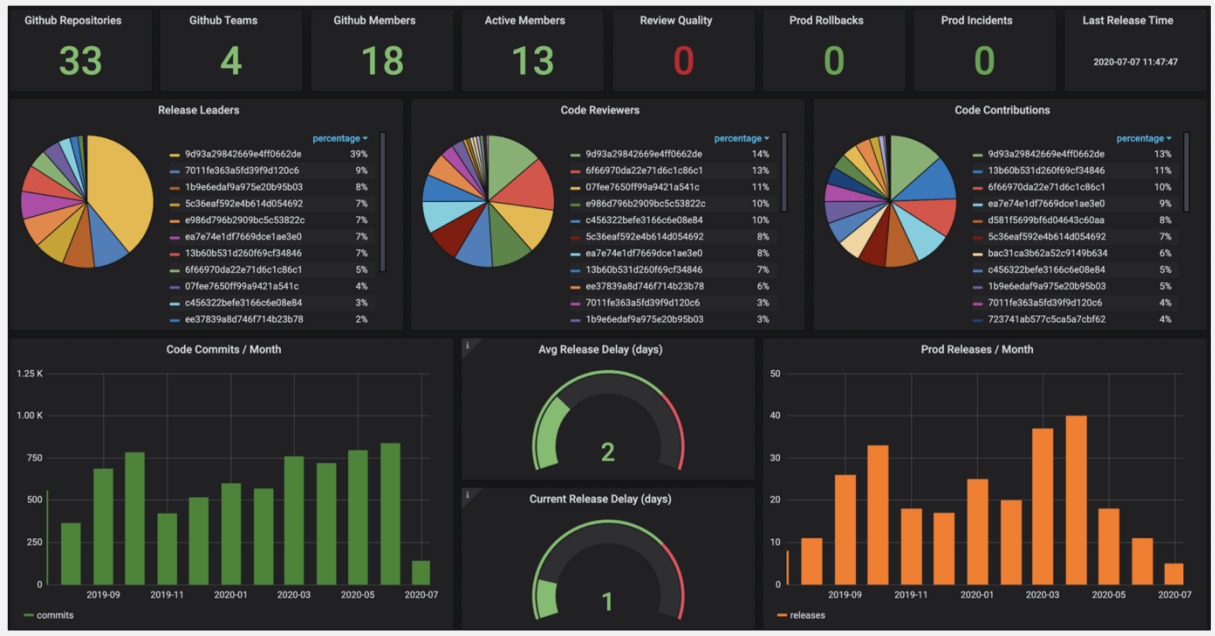
There are many aspects of an engineer's process which can be measured to produce data which definitively shows patterns and how an engineer works. If an engineer is active and
sticks to common practices like regular commits, unit testing, etc. it becomes very easy to gather data to assist in this process.



To measure an engineer's general activity, it would be useful to gathering as much time-oriented data as possible. Variables such as average monthly time spent committing and working on projects, average lifetime of a project, when an engineer began his/her career, etc. these can all be used to help convey how active an engineer is in general. A researcher could display data in a graphic form which shows how the engineer works with respect to time, something like total number of commits against time will show the acceleration of an engineer's work rate, the graph would be expected to be a steady slope if the engineer made a similar number of commits over multiple cycles of a specific time frame.
We could also retrieve several figures, such as number of projects, number of collaborative projects, connections (other engineers worked with), average change per commit, etc and just list these figures in infographic format. This would tell a lot about an engineer and how efficient/dedicated they are, very useful when trying to find a candidate for projects or to work with and useful to gather data for statistical purposes. Another big aspect we can use data to

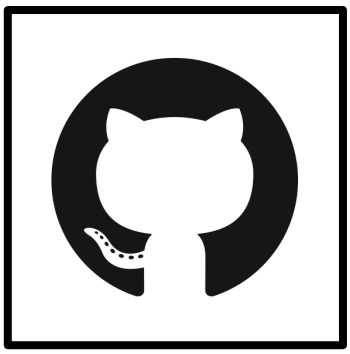
determine is how well an engineer works in a group, looking at things like commits, pulls, additions in comparison to other members of the group etc. Again, this can all be useful information when determining if an engineer is right for a job or a good fit for your development team.

Another aspect to look at when measuring engineering activity is the code itself. One could develop an algorithm to determine the efficiency or complexity of a body of code, which could be used to calculate some form of "score" for example, displayed to give a sense of the engineer's proficiency in coding. The languages used by the engineer could also be displayed in the form of a pie chart for example, this could show an engineer's preferred language or which languages they have more experience in.

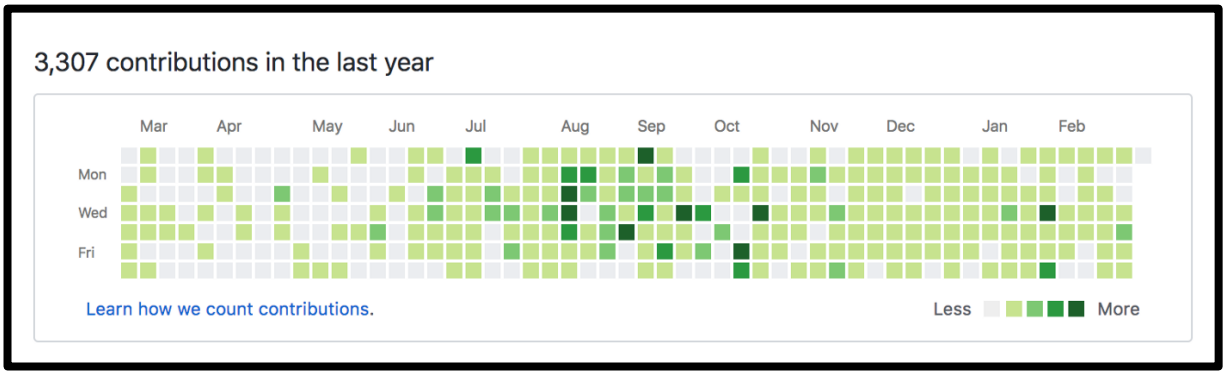


SECTION 2 – PLATFORM FOR MEASURING DATA

There has been an increasing demand for data on software engineers over the past few decades as the information technology industry grew at an exponential rate. This has caused companies and developers to come out with an endless list of platforms and applications to help capture and visualize these metrics. These can be useful for project managers, for example, to pick out candidates to take part in projects and ensure teams are on track for delivery dates.

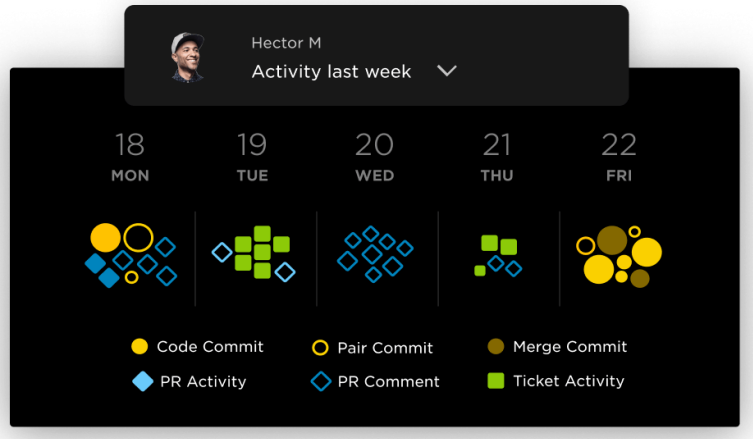


Software like Git and its more user friendly offspring GitHub allow engineers to manage and maintain development progress through tracking of each commit to the code by each member of a development team and assisting in merging of two different versions of the code. Version control software like these often allow us to collect data on engineers, for example, GitHub has a REST API which allows users to make requests to the GitHub servers and retrieve data on a plethora of information surrounding the users, projects and essentially any aspect of the development process.

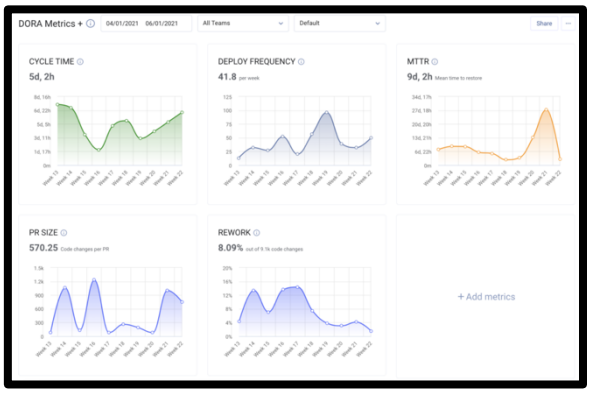


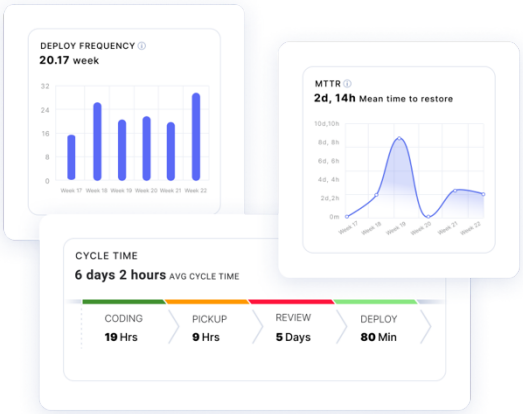
Git is probably the most widely used software when it comes to software engineering and has become an industry standard over the last few years, the complexity of the software is the reason a lot of professionals see fluency in Git as a standalone technical skill.

GitPrime (known now as Pluralsight) is a platform used to collect and display data from a git host. One of the advantages of Pluralsight Flow is it uses the information from pull requests linked to commits which would produce some insightful graphs.



Another software that is widely used in unison with Git and other collaborative repository platforms to analyze repositories is called LienarB. It has both free and paid versions which come with a different list of features. It can be used to analyze things like development time, deployment time and deployment frequency. Many companies and employees in positions like head of development will find this very useful for deciding deadlines, managing a team, or ensuring a smooth product roll out.



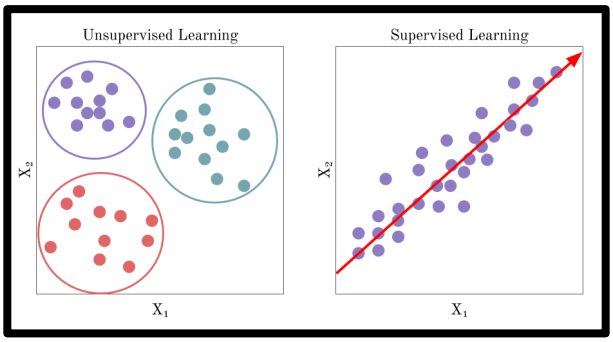


SonarQube is a free opensource software used to continually inspect the quality of a body of code, supporting more then 25 languages. It can check for bugs, redundancy and provides code coverage analysis. It displays all this information in a user friendly interface, easily understandable for a first time user.

SECTION 3 – COMPUTATIONS OVER SOFTWARE ENGINEER DATA

A widely used computational approach, to analyse the data gathered, that involves a regression model based on the number of lines of code is Constructive Cost Modelling. This is often used as a process of predicting parameters like effort, time, cost and size of software engineering projects and is known as a procedural cost estimate model. There are three categories, based on size, that use this model, organic, embedded and semi-detached. Projects with around 2,000 to 50,000 lines of code which belong to a small team with good knowledge of the domain are classified as organic. Projects with around 50,000 to 300,000 lines of code, ran by a medium sized team with mixed experience to deal with less familiar environments are known as semi-detached. Any project with above 300,000 lines of code, with experienced developers dealing with completely new and unfamiliar environments are labelled embedded.

A technology used across the board to analyse large sets of data is machine learning. Machine learning has proven to be incredibly accurate hence top class data analysist use it on the regular to help formulate results from data too lengthy to analysis using simple computational algorithms. There are three categories of machine learning, supervised learning, unsupervised learning, and reinforcement learning. It is categories based on how the model is trained. Supervised learning is when a model is trained with a dataset that is simply labelled, for example a large data set of engineer activity where each entry is labelled either “active” or “not active” based on the amount of commits for example. Once the model is trained using the given data set, we can run a new entry through the model and it will tell us if the engineer is active or not with a given margin for error. Unsupervised learning is used when we have unlabelled datasets, in this case the model deciphers its own correlating factors through exploration. The two fundamental types of unsupervised learning methods



are clustering and density estimation. The former (which is probably the most commonly used) involves problems where we need to group the data into specific categories (known as clusters) while the latter involves summarizing the distribution of the data. Reinforcement learning, while similar to unsupervised learning as the data analysed is not labelled, but in this case the dataset has a Boolean feedback. A few example of this would be scenario-based learning policies for highways, trajectory optimization or advanced path-finding algorithms. These methods could be applied to a large dataset of engineer activity collected via one of the methods discussed previously to uncover or predict certain elements of future entries acquired.

SECTION 4 – ETHICAL CONCERNS

The process of collecting and analyzing data to exploit and select candidates based on previous work, is always going to raise concerns surrounding ethics. The real parameter that it comes down to is what the data is being used for and whether the candidate consents to the data being shared.

The process of an employer collecting and analyzing data on code written by a team contracted to deliver a project and using the data for deadline prediction, future decisions and project feedback is hard to label "unethical" as the team members would have agreed to such terms in their contract. In terms of privacy this seems a fair practice as if you are good at your job, it would only benefit you for your employer to know you were good and if you are doing a poor job the employer has a right to know as your superior. Where the controversy comes into play is when companies or employers collect data on engineers without their knowing. The number one monitoring practice that is considered unethical, and in most cases even illegal, is monitoring employees without their knowledge or consent. This practice is considered legal when employers are suspecting malpractice, and want to catch employees red-handed. However, if companies simply want to keep an eye on their employees without telling them, they could face serious consequences. This raises the question, what about monitoring potential candidates?

The reason, in most cases, monitoring potential candidates would not be considered unethical since the means of retrieving the data is generally through some form of publicly available means. For example most employers would use GitHub accounts or other public accounts to recruit candidates for a job, where these accounts are often provided by the person in question themselves. In other cases this can be found quite easily if the candidate has a good online presence. However, even if this is the case and the data is found without the candidates knowing, it was said candidate who gave consent for this to take place when making a public repository account or public project portfolio. Although this is the case for most, some young engineers may not know the importance or impact of setting up a GitHub for example, early developers may not see the need for technologies like it and never set one up, causing them to lose out without even knowing. So perhaps a candidate for a given job has the capability to do the task to a very high standard and fits the team perfectly but will be disregarded immediately because they don't have any public repositories to show. Or maybe a candidate set one up and not use it seriously, taking away from any good work the engineer does in the future.

CONCLUSION

The topic discussed in this paper is a complex one, with many different aspects and subtopics, software engineer monitoring is an extremely broad topic. There are countless measurements we can take to reveal different things about engineers and how they work. The demand for data on software engineers has caused a plethora of software's and platforms to be developed which help collect and display data in a user friendly and understandable way. The ethics of all of this is a controversial topic as is often the case when it comes to ethics, one which can be cleared up by breaking the software engineering community into subgroups and specifying the circumstances of why a group or individual is being monitored. Software engineer monitoring is a process which can help better a developer's skills or jeopardies one's career.

REFERENCES

https://www.sonarqube.org/

https://www.atlassian.com/software/jira

https://towardsdatascience.com/supervised-vs-unsupervised-learning-bf2eab13f288#:~:text=Some%20examples%20of%20unsupervised%20learning,Component%20Analysis%20and%20Hierarchical%20Clustering.

https://www.geeksforgeeks.org/measuring-software-quality-using-quality-metrics/

https://www.entrepreneur.com/article/356544

https://en.wikipedia.org/wiki/Software_metric#Common_software_measurements

http://www0.cs.ucl.ac.uk/staff/mharman/ieee-computer.pdf