MEASURING SOFTWARE ENGINEERING

TASK

"To deliver a report that considers the ways in which the software engineering process can be measured and assessed in terms of measurable data, an overview of the computational platforms available to perform this work, the algorithmic approaches available, and the ethics concerns surrounding this kind of analytics."

INTRODUCTION

For the longest time the best way to measure the progress of a software engineer has been disputed. Some see this as being the biggest issue with the industry – how do we measure such a complex process? How can we tell if any progress is being made? In my opinion, however, this problem can be easily tackled and I plan on proving this through the writing of this report and the exploration of the different methods through which we can in fact measure the software engineering process. These methods can be divided into four groups:

1. Measurable data
2. Computational platforms
3. Algorithmic approaches
4. Ethics

THE SOFTWARE ENGINEERING PROCESS

Before beginning such a discussion, however, it is important that we understand the software engineering process itself.

FIGURE 1: DIAGRAM OF THE SOFTWARE ENGINEERING PROCESS

According to IEEE, software engineering is "*the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software*". The typical software engineering process can be seen as having five individual stages.

1. Analyse Users Requirements
   At this stage the software engineer will carry out research as to why they are producing such a product? What do the users wish to gain through the use of this product? How will this product benefit the user? It is necessary for the engineer to carry out some degree of market research to ensure that they are meeting the users' requirements and that they understand what form of input the users will have and what type of output they desire.
2. Design the Program
   The software is now equipped the knowledge what their users demand from the product and so they can get to work designing the product itself. This involves constructing a blueprint of the software and discussing the constraints placed on the product. This blueprint should include a low-spec description of the design of the individual components of the software and how they will interact together.
3. Build the System
   This phase simply involves building the software based on the research that has been carried out and the original design that has been decided upon. The engineers write code to bring their designs to life.
4. Document and Test the System
   When the code written by the engineer has been integrated, it is necessary that they test the software. They must check for errors and bugs in their program. The engineer may then invite a sample of potential users of the software to do a trial run of the software. These users can provide valuable information for the engineer. The will inform the engineer whether or not they have met the users' requirements and if the product is indeed user friendly. This is one of the most important stages of the process.
5. Operate and Maintain System
   At this point the system is ready for implementation. The people who desired the software now have it available to them, however, the engineer's task is not complete. It is now up to the engineer to monitor the system and to adapt it to the users' ever changing expectations and requirements. Maintenance is required to ensure that the software overcomes any new issues it may encounter.

Evidently, the software engineering process is long and complex, consisting of many stages and constant work for the engineer, ensuring that every phase of the process is carried out successfully and efficiently. Still the question remains, how do we measure this process? I will now try to answer this question through the discussion of four main topics – measurable data, computational platforms, algorithmic approaches and ethics.

Measurable Data

The choice of which metrics to use when measuring the work of a software engineer remains relatively unclear to this day. Software engineers work differently to employees in almost every other industry in that their productivity cannot be gauged by the number of units they produce or the amount of hours they work. Their complex work must be measured through trends and comparisons. The paper "Collecting, Integrating and Analysing Software Metrics and Personal Software Process Data", by Sillitti, Janes, Succi and Vernazza, begins by stating that there are "*two kinds of measures in engineering: product metrics and process metrics*."

Process Metrics

Process metrics involve the software engineering process itself. They allow managers to analyse the status of a project, examine the potential risks faced by a project and to make vital decisions with regards to workflow and the individual tasks involved in the process. Such metrics include the following:

1. Lead Time
   The lead time of a project is the time between the start of the project and the end of that same project, when a finished product is made available to customers. By examining past lead times of a particular development team, one can make educated estimations as to when the project at hand will be completed. The shorter the lead time the better as it shows that the team is working efficiently and doing their best to ensure the prompt delivery of the product.

2. Cycle Time
   This can be defined as the tome it takes a development team to edit and adapt the software system they are producing and to deliver that change in production.

3. Deployment Frequency
   This form of metric involves analysing how often a team makes deployments. Most managers prefer teams to make smaller deployments more regularly. If a deployment is small, it is easier to test and therefore, the software can be improved upon and made as efficient as possible.

4. Open/Close Rates

With every software development project there are production issues. Open/close rates tell us how many of these issues are reported and managed during a particular amount of time. The important thing about this figure is that it should be improving as time goes on.

5. Commit Frequency
   Managers may track the number of commits being made to repository daily and the frequency of these commits in order to gauge the productivity of their team. The more commits the better as it shows that the team are working constantly and sharing their work with each other to further along the production of the software.

6. Pull Requests
   When it comes to pull requests, there is more than one metric that should be of interest to managers:
   - The number made per week.
   - The number merged per week.
   - The average time taken for pull requests to merge.

Product Metrics

Product metrics involve measuring the product itself. These metrics allow the engineer to better understand the different aspects of the model and the quality of the software produced. These metrics are also known as quality metrics. They include the following:

1. Mean Time between Failure
   This refers to the average time between failures of the program. The shorter this time is, the higher the quality of the software.

2. Mean Time to Repair
   This refers to the average time taken to fix the program once it has failed. The shorter this time is, the better the program.

3. Application Crash Rate
   This metric is related to the two above. It calculated by dividing the number of times the application fails by the number of times it was used.

4. Defect Removal Efficiency
   This metric calculates the number of defects found by the user of the software after product delivery in relation to the errors detected before product delivery.

5. Test Coverage Ratio

This refers to the ratio between the total lines of code written and the number of lines of code the test cases execute. The higher this number is, the better. Generally, it lies around 80%.

Software engineers are capable of carrying out both product and process metrics of their own work by using the *Personal Software Process*.

This process consists of three stages: planning, development and post-mortem. In the planning stage the engineer determines the required functionality of the program and writes a task report based on their findings. The development stage involves the engineer writing code to develop a software that fulfils the requirements determined in the first stage. They then review and test and edit this code to the best of their abilities. During the post-mortem stage the engineer once again writes a task report and compares the efficiency of the software produced to the estimated efficiency of the desired outcome. They should also calculate running times and defect rates of their code.

COMPUTATIONAL PLATFORMS

Data analytics has become a major industry over the last decade. Nowadays, there are many companies that have been set up to analyse and measure the software engineering process. In their paper, *Software Analytics in Process* (2013), Dongmei Zhang and Tao Xie state that "*software analytics is to enable software practitioners to perform data exploration and analysis to obtain insightful and actionable information for data-driven tasks around software and services*". The following companies strive to carry out such software analytics:

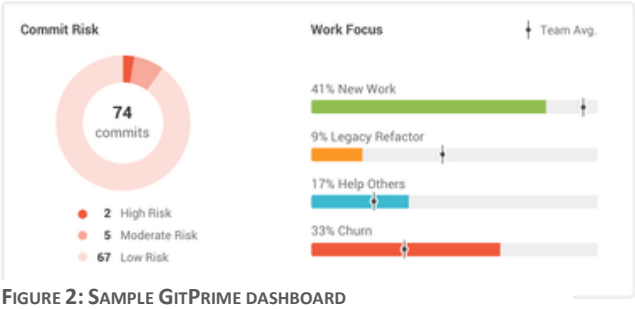

FIGURE 2: SAMPLE GITPRIME DASHBOARD

1. GitPrime

Many modern coders make use of Git-based repositories, such as GitHub and Bitbucket, through which to do their work. GitPrime pulls data from these repositories and transforms this data into reports that can be easily understood and analysed so that managers can gauge how their employees are performing. Through the use of this tool a team leader can gain invaluable visibility into the performance of their team. As a result, it is much easier for a manager to track trends in their employees' performance and efficiency. The manager can also detect if some employees are working too much and are likely to run out steam, improving these employees' work-life balance. Metrics outputted by GitPrime include:

- Impact: The impact a change has on a project and the difficulty with which these changes are made.
- Churn: How much the code written has been edited.
- TT100 Raw: How long it takes an engineer to write 100 lines of code.

- TT100 Productive: How long it takes an engineer to write 100 lines of code and to churn this code.

2. Velocity

Velocity is similar to GitPrime in that it extracts data from a Git-based repository to carry out analysis on a team of engineers. Velocity focuses mainly on pull requests to measure these engineers. From this analysis, a team manager gains an insight into their team's efficiency and how this improving over time and also on their workflow, detecting certain areas where the work might be getting stuck. Metrics outputted by Velocity include:

- Impact
- Rework: How much an engineer reworks code that they have written.
- Pull Request Activity Level: This metric gauges how much your team is focusing on one particular pull request.
- Review Cycles: The number of times a pull request goes between the contributor of a piece of code and the reviewer of this code.

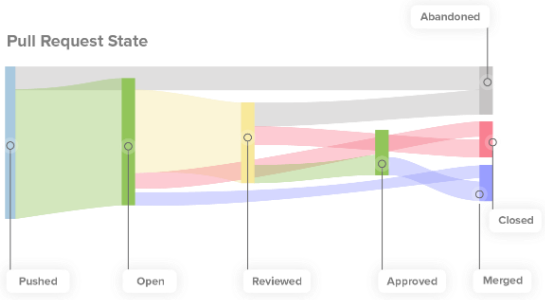

FIGURE 3: VELOCITY ANALYSIS OF A PULL REQUEST

3. Squore

Squore is another major software analytics company. Squore provides invaluable information with regards to the quality of a software project and the performance of the team producing this software. Squore employs automated data processing to assess software based on several Key Performance Indicators. Among these KPIs are: code cloning, complexity, rule compliance, self-descriptiveness and innovation rate. Squore then compiles a series of dashboard reports to display the information that it has gathered allowing managers to make optimised and informed decisions.

FIGURE 4:SQUORE DASHBOARD

4. Waydev

Waydev is akin to GitPrime and Velocity, using Git-based repositories to calculate a number of metrics. These metrics include:

- Impact
- Commits per day: How much a team pushes code on a daily basis.
- Code churn
- Efficiency: The efficiency of a single engineer or of a team of engineers.
- Active days: The number of days that engineers have contributed to a project in some form.

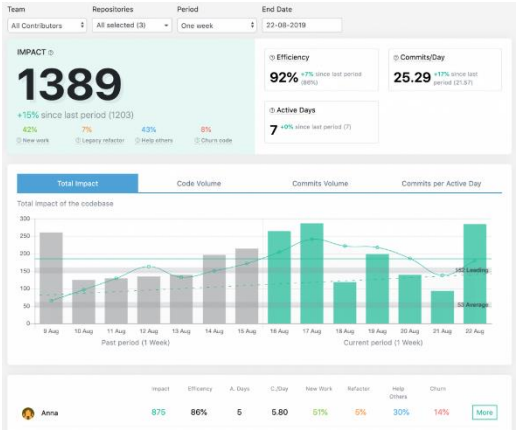

FIGURE 5: WAYDEV DASHBOARD

Waydev displays this information through the use of dashboards. Managers interpret these dashboards to determine how individual employees are performing and how a team is performing together. Waydev is a cheaper alternative to GitPrime and Velocity as it cannot be used over the same number of platforms.

5. Code Time
   Code Time is an open source plugin that provides programming metrics in a code editor such as Atom or Eclipse. This plugin provides the engineer themselves with daily reports of their coding activity in the IDE that they are using. The engineer is also sent a weekly email to document their programming that week. Users may also log into the Code Time website to see illustrations of the date collected by Code Time. Other stats provided by Code Time include the engineer's best music for coding and the speed of their commits.

ALGORITHMIC APPROACHES

The software analysis tools listed above make use of numerous algorithms in order to carry out their analysis. It would be extremely cumbersome and challenging to analyse a large quantity of data manually. Algorithms analyse this data more efficiently and provide more information and recognise particular trends and patterns in the data. The majority of these algorithms make use of machine learning. Machine learning is a method of data analysis that automates analytical model building. This form of analysis is a branch of AI which believes that systems can learn from data and identify patterns in this data. Machine learning can be broken down into three main methods:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

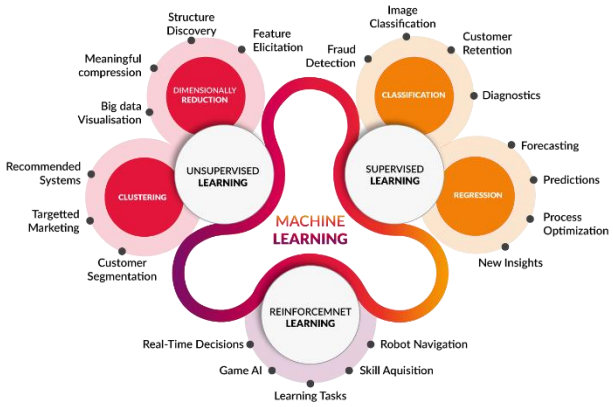

FIGURE 6: MACHINE LEARNING

SUPERVISED LEARNING

Supervised learning is a type of system in which both input and the desired output data have been provided. The dataset being examined has known inputs and outputs. The task at hand is to determine how these inputs and outputs are linked to each other. The algorithm iteratively makes predictions based on the dataset. If these predictions are wrong, they are corrected. This process is then repeated until the algorithm reaches an acceptable level of performance and accuracy. Supervised learning is typically divided into two categories: classification and linear regression.

1. Classification
   Classification algorithms are used to make predictions when the output provided is categorical. The data may be bi-class or multi-class. These algorithms include linear classifiers, nearest neighbour, support vector machines, decision trees, boosted trees, random forest and neural networks.
2. Linear Regression
   Linear regression algorithms are used to make predictions when the output provided is continuous. This algorithm attempts to model the relationship between an input and an output by fitting a linear equation to the observed data.

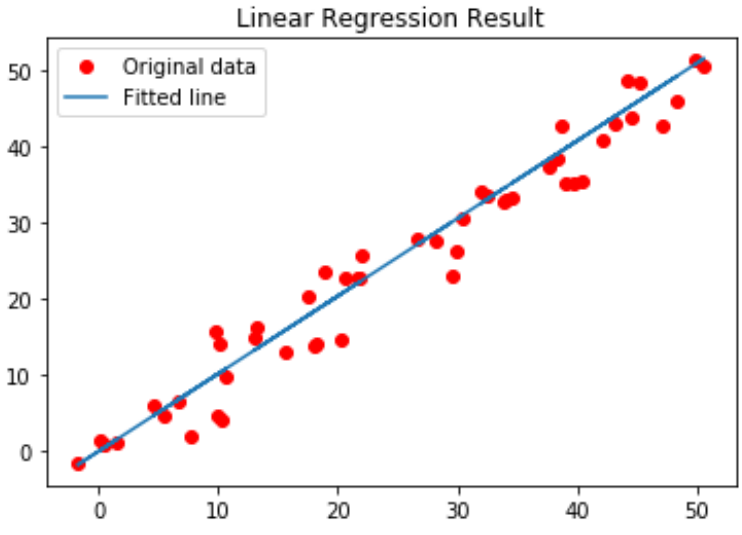

FIGURE 7: LINEAR REGRESSION

Unsupervised Learning

Unsupervised learning algorithms are algorithms for which only input data is provided, the output data is unknown. The task at hand in this case is to model the structure of the data and make inferences about the data. There are two categories of unsupervised learning algorithms: clustering algorithms and association rule algorithms.

1. Clustering Algorithms
   Clustering involves splitting data into groups according to similarity. The risk with clustering algorithms is that they may overestimate the similarities between clusters and not treat each data point as an individual point. Clustering algorithms include k-means clustering, mean-shift clustering and agglomerative hierarchical clustering.

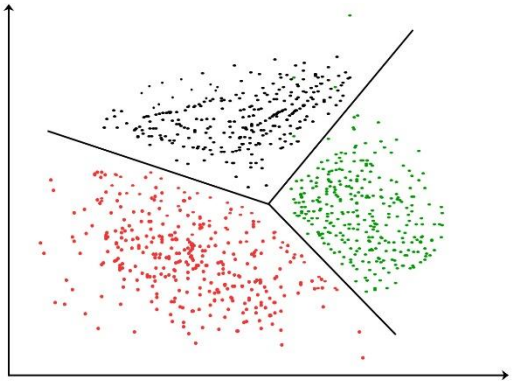

Figure 8: Clustering

2. Association Rule Algorithms
   Association rule algorithms determine interesting relationships between different data points in a data set. Association rule algorithms include the Apriori algorithm and the Eclat algorithm.

Reinforcement Learning

Reinforcement learning is a form of dynamic programming that trains algorithms using a system of reward and punishment. These algorithms learn by interacting with their environments. If the algorithm performs correctly it is rewarded, if not it is punished. The goal is for the algorithm to maximise its reward and minimise its punishment. Reinforcement learning algorithms are centred around decision making. This division of machine learning is becoming more and more prominent in today's world with the impending arrival of driverless cars.

Overtime, the accuracy of the decisions made by machines will improve exponentially. It is important to remember, however, that these algorithms lack a human element. Therefore, they should be used in conjunction with human judgement when measuring the performance of an engineer. Someday this intervention may not be necessary but we are not yet at that stage.

Ethics

One of the most prominent issues in our modern society is that of illegal data mining. In 2018, for example, Facebook's stock price plummeted when it was revealed that Cambridge Analytica had harvested the personal data of millions of Facebook users. Therefore, after this point it is necessary to question if the collection of this amount of data from software engineers is ethical. Is the data being collected public or private? How much data is too much data?

The sale of data is of major concern when discussing the ethics of data collection. Data can provide companies with valuable information regarding their target market and customers. This information includes trends in society, the best social medium through which they should advertise and customer expectations. These companies believe that data is an asset capable of improving their performance. Data brokers collect information about customers and then sell this information onto other companies. This industry is a lucrative but unethical one. The Cambridge Analytica and Facebook scandal is an excellent example of the sale of data. This scandal tainted Facebook's reputation as a trusted social media platform after taking advantage of their users.

Transparency is essential when analysing the ethics regarding data collection. The software engineers whose work is being measured must be kept informed of what data is being collected from their work and what analysis is being carried out on this data. There should be a constant stream of communication between managers and employees broadcasting both information and feedback from the process. I recently listened to a podcast in which Joe Rogan spoke to the famous American whistle-blower Edward Snowden. Snowden is currently under political asylum in Russia after he divulged that the American government was hoarding private information from American citizens through the National Security Agency's PRISM program. Unsurprisingly, this caused uproar in the US as Americans began to believe that they could not trust their own government. Many technology companies were participants in the program, giving the government many avenues through which to gather data on private citizens. These companies include Microsoft, Google, Facebook, YouTube and Apple. This lack of respect for the privacy of people is shameful. It is entirely unethical, highlighting the major need for transparency in data collection.

We cannot explore the ethics of data collection without discussing the legal system and data sovereignty. Data sovereignty is the concept that information which has been converted and stored in digital form is subject to the laws of the country in which it is located. Currently the major concerns surrounding data sovereignty include the enforcement of privacy regulations. This issue has become more prominent with the ever-increasing capabilities of cloud computing. In response to this, many countries have regulated new compliance requirements by adapting their current laws so that customer data must be kept within the country where the customer resides. In 2018, the EU parliament introduced the General Data Protection Regulation (GDPR). The main purpose of this law is to harmonise data privacy laws across Europe and to reassess how organisations in Europe approach data privacy. If companies fail to abide by the new GDPR regulations, they face major fines. This calls into question the ethics of collecting the volume of data needed to measure the

software engineering profession. Clearly defined boundaries, laws and legislation such as the GDPR are essential.

Conclusion

I believe that is indeed possible to measure the work of a software engineer. There are many approaches to this practice that prove to be extremely effective in gauging the performance of these hardworking engineers. There are also many available platforms through which one can measure the software engineering process and the effort being put into this process by the members of a team of engineers. The algorithms being used in the background by these platforms are extremely effecting and through the different departments of machine learning the algorithms are becoming more accurate in their measurements every day. However, the practice of data collection can be quite controversial. Members of our modern society enjoy sharing aspects of their lives with others through many mediums, however, when it comes to data that we wish to keep private we are more determined than ever to do so. Therefore, in order to measure the work of a software engineer through the practices above clarity and transparency are of the utmost importance. The metrics being measured and the reasons why they are should be established and agreed upon with the engineers prior to the collection of any data from them. If managers fail to do so they risk lawsuits and waste valuable time and resources. Therefore, before deciding how they will measure their employees, managers must question the ethics regarding each possible option.

BIBLIOGRAPHY

- https://anaxi.com/software-engineering-metrics-an-advanced-guide/
- "Collecting, integrating and analysing software metrics and personal software process data" – A. Sillitti, A. Janes, G. Succi and T. Vernazza
- https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams
- http://www.sigada.org/conf/sigada2001/private/SIGAda2001-CDROM/SIGAda2000-Tutorials/SF2-Roy-Part1.pdf
- "Software Analytics in Progress" – Dongmei Zhang, Tao Xie
- https://www.producthunt.com/posts/gitprime/reviews
- https://www.gitprime.com
- https://codeclimate.com/blog/velocity-vs-gitprime/
- https://www.getvelocity.co/features
- https://www.squoring.com/en/produits/squore-software-analytics/
- https://waydev.co/
- https://www.software.com
- https://www.sas.com/en_ie/insights/analytics/machine-learning.html
- https://medium.com/machine-learning-researcher/association-rule-apriori-and-eclat-algorithm-4e963fa972a4
- https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68
- https://towardsdatascience.com/what-are-supervised-and-unsupervised-learning-in-machine-learning-dc76bd67795d
- https://eugdpr.org
- https://whatis.techtarget.com/definition/data-sovereignty
- https://en.wikipedia.org/wiki/PRISM_(surveillance_program)
- https://www.wired.com/story/after-six-years-in-exile-edward-snowden-explains-himself/