Measuring Engineering - A Report

Contents

Introduction:

In this report I will discuss the different ways in which software engineering processes can be measured and assessed.

In order to understand how software engineering is measured and assessed we must first get a good understanding of what it is. Software engineering is a direct sub field of engineering and hence software engineers must apply the principles of engineering in order to design, construct and test end user applications that satisfy the user's needs. Essentially software engineers determine how users engage with technology. In the last 50 years the discipline has experienced massive growth. This growth has brought the need for proper measuring and assessing the process of software engineering. In this report I will analyse how this process is measured in terms of:

1. Measurable data.
2. Overview of the computational platforms available to perform this work.
3. Algorithmic approaches available.
4. Ethics concerns surrounding this kind of analytics.

In order to fully understand how software engineering is analysed, I will first look into the current software engineering process. It is important to understand the stages of development which each metrics investigates.

Software Engineering Process:

A software development process, also known as the software development life cycle (SDLC) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from scratch or modifying an existing system. All software processes include the following four activities:

1. **Software specification** : Define the main functionalities of the software and the constraints around them.

2. S**oftware design and implementation**: The software is to be designed and programmed.
3. **Software verification and validation**: The software must conforms to its specification and meets the customer needs.
4. **Software maintenance**: The software is being modified to meet customer and market requirement changes.

1. Measurable Data

In the early days of measuring software engineering processes one of the biggest issues was how to collect data and what data to collect. At this time data was collected by manually inputting data into spreadsheets. This led to very little data being collected as developers did not see the importance of this data collection and would rather spend their time developing software. Also, the data that was collected was prone to human error.

In time however the importance of data to monitor work and help make improvements was realized. Data can help companies minimise wasted time and costs and maximise efficiency and productivity. Recent decades have seen rapid growth in data from communication, information and technology systems. The increase in data is a direct by-product of our growing use of technology.

Using this data to measure software is not an exact science as there is no single method in which we can rate production levels. The production level also changes depending on which stage of development the software is in. Measuring processes is both challenging and vitally important for software engineers. There are 3 categories of metrics required to measure software engineering processes: productivity metrics, process metrics and quality metrics. I will give examples of each below.

But first what is a software metric? A software metric is where data from an application development lifecycle is measured and is used to measure a software developers productivity. It can originate from a single or multiple data source

Productivity metrics

Software productivity can be defined as the ratio between the functional values of software produced to the effort and expense required for development.

a) **Code churn** is the percentage of a developer's own code representing an edit to their own recent work. To compute it we measure the lines of code that were modified, added and deleted over a short period of time such as a few weeks, divided by the total number of lines of code added. Code churn is considered to be 'non-productive work' but high churn may be attributed to other aspects such as: unclear requirements, indecisive stakeholders and a difficult problem to solve. Code churn is best utilised when a baseline for a developer of development team is established and action is taken when the churn deviates from this baseline.

Fig. 1 Code churn chart.

a) **Sprint burndown.** Instead of considering a project as a number of tasks and that all tasks to be done are equal, teams can consider story points. A story point is a high-level estimation of complexity in a software feature. Sprint burndown communicates the complexion of work throughout the sprint based on story points.

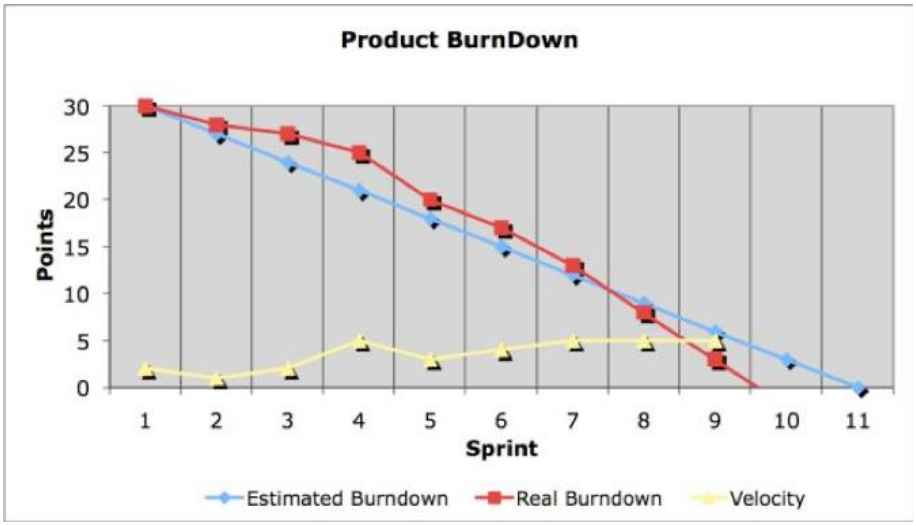

Fig 2. Spirt burndown chart. (Scrum Institute 2019)

Process metrics

Process metrics show performance of software engineers processes and software development workflow.

a) **Lead time** is the difference between the initiation and the completion of a process. In terms of software engineering it is the quantity of time elapsed between the identification of a requirement and its fulfilment. Lead time is a good way of tracking how responsive the software developer is to their customers.

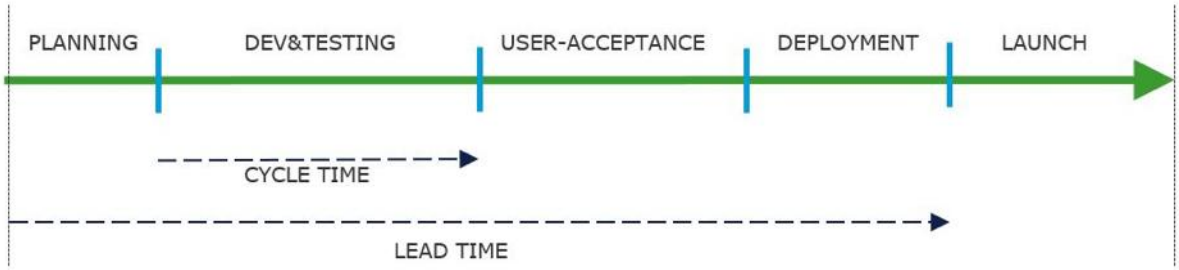

Fig.3 Lead time chart (J. Ovstass 2018)

b) **Number of commits** is a measurement of the amount of time a software engineer spends on their code. This is a good way of measuring activity levels in software engineers but using this metric for productivity would be incorrect. Large number of commits does not coincide with better progress.

Fig 4. Github commit frequency table (Github 2019)

Quality metrics

Software quality may be defined as "conformance to requirements". Such "conformance" means, most generally, that the product meets the needs of the user and satisfies stated performance criteria. (Gaffney 1981)

a) **Test coverage ratio** is calculated by getting the ratio between the total lines of code in a piece of software and the number of lines of code currently used to test the code. A program with high test coverage, measured as a percentage, has had more of its source code executed during testing, which suggests it has a lower chance of containing undetected software bugs compared to a program with low test coverage. Generally, developers accept around 80% as sufficient test coverage.

b) **Application crash rate (ACR)** is calculated by dividing how many times an application fails by how many times it is used. There are a number of ways to assess this metric. First is app crashes per user and an acceptable range for this would be less than 1%. Second is app crashes per session an acceptable range here would be less than 0.1%. Lastly, there is an app crashes per screen view which has an acceptable range of less than 0.01%.

Limitations:

Managers cannot simply look at metrics and jump to conclusions, metrics cannot be used to assume causes. Time must be taken to discuss with developers involved in the process to uncover more about the story behind the numbers that the metrics produce.

We concluded that the existing models are incapable of predicting defects accurately using size and complexity metrics alone. Furthermore, these models offer no coherent explanation of how defect introduction and detection variables affect defect counts. (Fenton, N. E., and Martin, N. 1999)

2. Overview of the computational platforms:

With the quality and efficiency of code becoming increasingly significant in the industry of software development, large software development organizations must collect useful data to be able to build successful software whilst staying competitive. Many analytic services companies have sprung up in recent years to process this data. These companies supply software that will monitor the code being committed for bugs and security leaks among other things. I have previously discussed the metrics these companies use so now I will have a look at a few examples of these software development analytics companies.

a) Personal Software Process

PSP is a structured software development process that was designed to help improve software engineers performance by using a disciplined, data driven procedure. The PSP was created by Watts Humphery who taught through analysis of practical application of the process, the process should be changed for the better (W. Humphery 1996). Since it was first founded there have been several modifications to the process but they all require a lot of manual input from the engineer. Forms, time logs, project summaries, time estimates and design checklist all have to be filled out by the engineer. Collecting and reviewing all this data is awkward and time consuming. This form of manual data entry makes the PSP prone to human error which result in data quality problems also manual collection requires a long time for assessment. Despite this some people believe there is still use for the PSP. The PSP can be used to teach useful software engineering skills, despite its potential for data quality problems (P. Johnson 1998). However the PSP doesn't provide enough return on investment for it to have use outside of teaching.

b) Leap

Leap toolkit was developed by the University of Hawaii in order to address the quality problems with PSP by automating and normalising data analysis. Leap added in analysis not included in PSP such as regression analysis. Leap still required some form of human input and since its introduction to the industry it has become abundantly clear that there is no way to fully automate the PSP as manual input will be required, leaving it prone to human error.

c) Hackystat

Hackystat is a fully automated computational platform. Developers began using Hackystat by attaching sensors to their development tools. These sensors gathered process and product data and sent it to a server where other services were analysed. Some features of Hackystat include unobtrusive data collection (i.e users shouldn't notice their data being collected) and the ability to collect data from both the client side and server side. One of the platforms most interesting feature is it fine grain data collection this feature collects data in real time, tracking developers as they edit code. This feature is one of the main benefits of Hackystat however many software engineers find it intrusive and are uncomfortable with this level of data being collected.

The nature of this collection allows managers to have a deep understanding of the work that is being done, however, many developers were uncomfortable with the accessibility the platform gave others into their work. (Johnson, 2013).

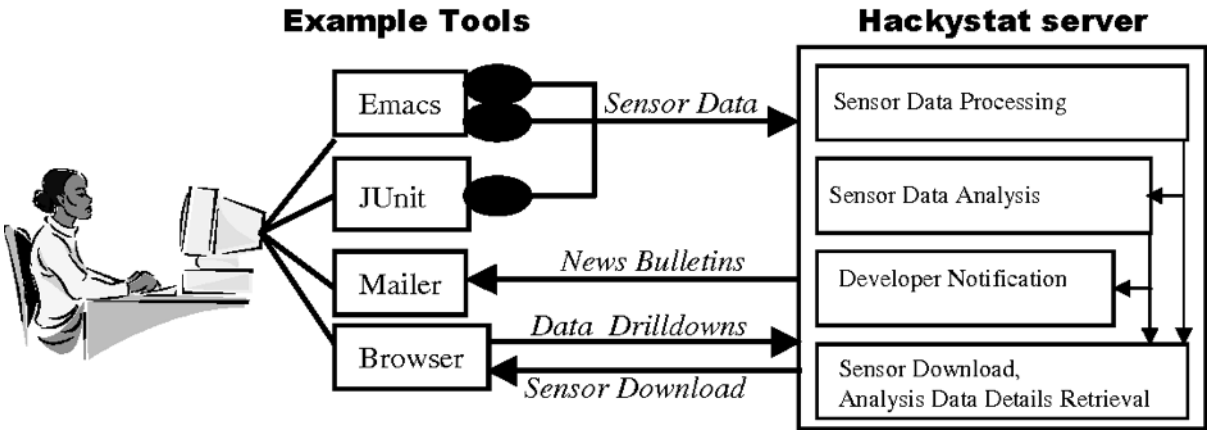

Figure 5. Basic communication pathways in Hackystat. (P. Johnston 2001)

d) Code climate

Code climate was founded in 2011 to help solve a problem co-founders Bryan Helmkamp and Noah Davis experienced as software engineers: how can you ensure software quality even as you move further away from the code. To solve this, they built an automated code review tool which has helped over 100,000 developers. Developer use code climate to make their code easier to understand, prevent repetitive code, improve the code reusability and ensure it is easily modified.

Code climate is used by several large companies such as Sage, New Relic and The BBC.

Some drawbacks that users report is that there is no detailed description of the issue, only a header with source code and that it doesn't support Objective-C. In terms of price it is one of the most expensive tools in comparison to its competitors. However, its wide range of supported languages, technologies and frameworks as well as its Github integration make it one of the leading software development analytical packages. Hence its extensive use in the software industry.

3. Algorithmic approaches:

For many companies, big data – massive volumes of raw structured, semi-structured and unstructured data- is an untapped resource of intelligence which can be utilized to enhance software development. Companies use algorithms to analyse this data continuously. There are a large number of these algorithms available but considering the importance of machine learning in this discipline I will focus mainly on this form of analysis.

Machine learning is becoming an incredibly powerful tool to make predictions or suggestions based on large amounts of data. Machine learning is used by companies such as Amazon and YouTube to make suggestions to their users based on previous purchases or video choices. But what is machine learning? Machine learning gives computers the ability to learn without being explicitly programmed (A. Samuel 1959). It is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence. There are many methods of machine learning, I will discuss the most popular types below.

Supervised learning:

Supervised learning is where there is an input variable and an output variable, and an algorithm is used to learn the mapping function from the input to the output. Then once the algorithm is mapping the function well enough it can be used to predict the outputs for new inputs. It gets its name from the fact that it is learning from a training data set, where the answers are known, so the algorithm can be corrected until it has reached an acceptable level of performance. Supervised machine learning can be split into Linear Regression -where a regression line is fit to the data, and from this formula can be derived so we can predict the output for variable 1 given we have the value of variable 2- and classification- where the computer learns a classification

system and can then process inputs into specific groups. Below I will give an example of a common supervised learning applications.

a) K-Nearest neighbour (KNN): KNN is a method used for classification. KNN is one of the simpler techniques used in machine learning. It is commonly used in the industry due to its simplicity and low calculation time. KNN will classify similar data points together in a test set, it will then use this test data to make an educated guess on what an unclassified point should be classed as. To do this accurately, KNN is very much reliant on the quality of the test data. KNN is often used in image recognition and recommendation systems. KNN is the technology Netflix uses to recommend similar movies to those you have previously watched. For accurate prediction it is important to choose an optimal value for K. The lower the value of K the lower the value of bias -the difference between the true label and our prediction- in the model. If we increase K, the bias will decrease, however this will cause a higher chance of an error in our test data, which causes high variance. As K increases, the number of neighbours and the complexity of the model also increases.

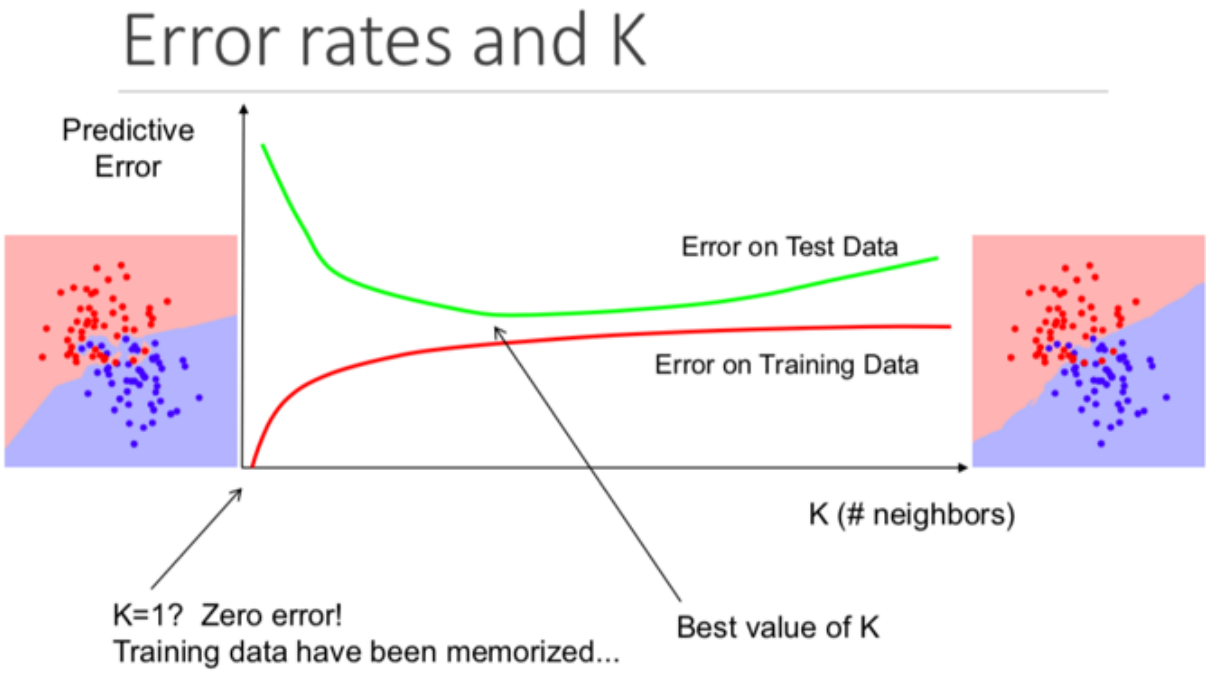

Fig. 6 Error rate and K. (S. Singh 2017)

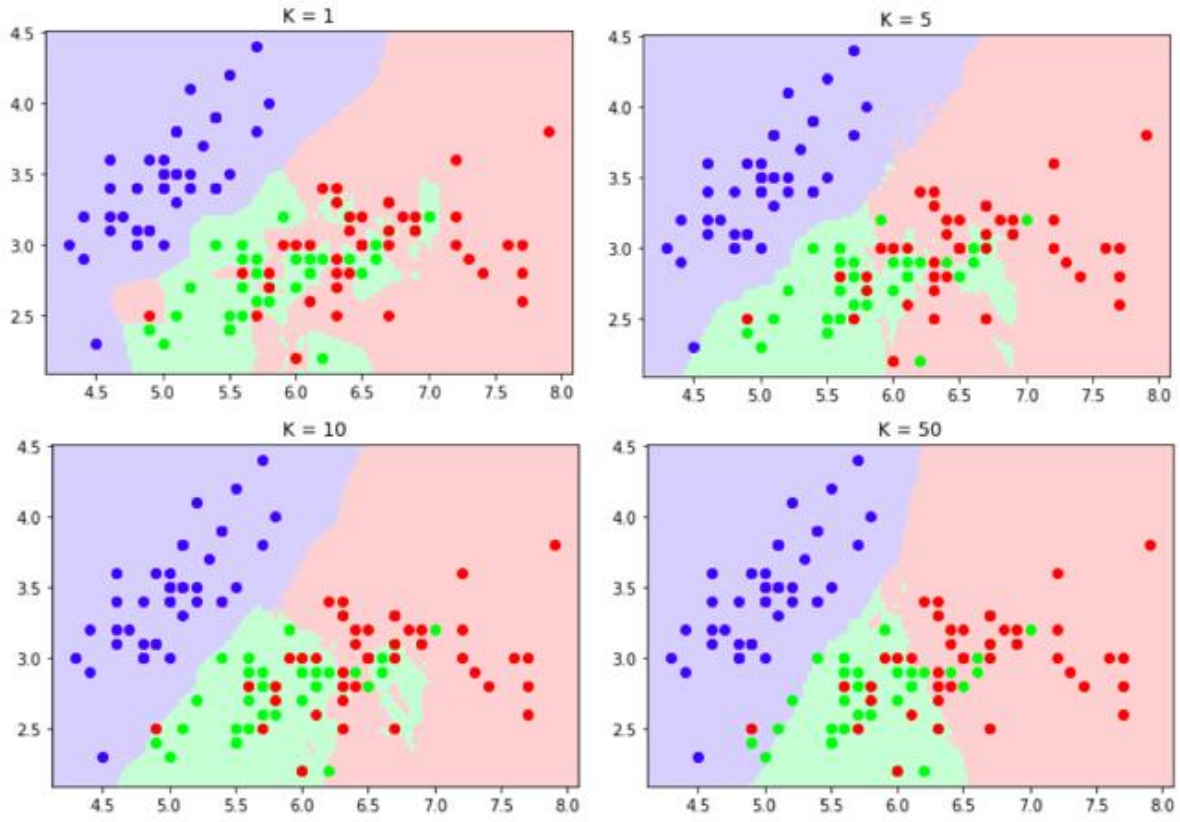

Fig. 7 Visualization of different values of K. (Medium 2018)

Above we can see that when K is small there are some outliners of green and red that are still in the boundary of green and red. When K becomes larger the boundary is more consistent and reasonable.

Unsupervised learning:

Unsupervised learning is where there is only input data and no corresponding output variables. The goal is for unsupervised models to explore the data and find some structure within the data in order to learn more about the data. Unsupervised learning gets its name from the fact that they are left to their own devices to explore the data. Unsupervised learning is effective on transactional data. It can be used to identify distributions of customers with similar preferences who can be treated similarly in marketing campaigns. There are two main types of unsupervised learning problems: clustering- this is where you want to discover groupings within data, such as grouping customers by purchasing behaviour- and association- this is where you are looking to try find rules that describe portions of the data, such as employees who are good at X are usually also good at Y. Below I will give an example of a commonly used unsupervised learning algorithm.

a) **K-Means Clustering:** K-Means Clustering is considered one of the simplest and most popular unsupervised algorithms. The objective is to group similar data points together and discover underlying patterns. To do this K-means must find a fixed number of clusters in a dataset. A cluster is a collection of data points with specific similarities. To implement K-Means clustering you assign K random centroids throughout the data. For every data point measure which centroid it is nearest to and assign the data point to this centroid. For every centroid, move the centroid to the average of the points assigned to that centroid. Repeat the last three steps until the centroid assignment no longer changes. The algorithm is said to have “converged” once there are no more changes.

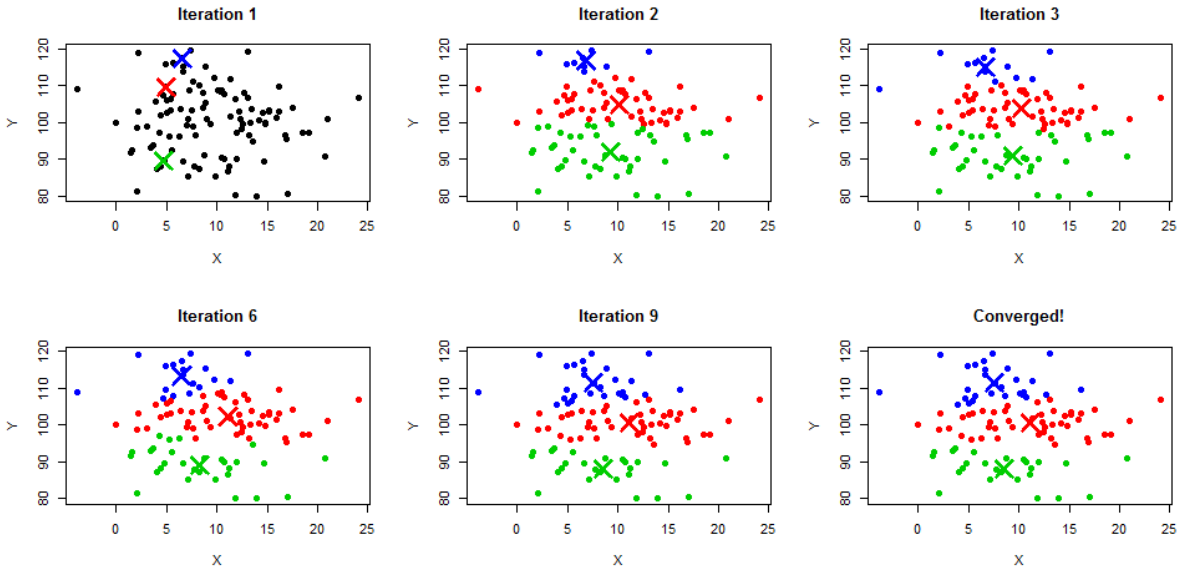

Fig 8. Iterations of K-Means Clustering (Learn By Marketing, 2019)

4. Ethical concerns:

Many ethical concerns arise in this area of data collection. At the moment data collection and analysing is a new and complex issue. Frameworks are yet to be put in place and until they are the line between acceptable levels of information to collect and intrusion of privacy is been left to those collecting and analysing to decide. As trends like machine learning, big data and data analytics become more and more prevalent in our society, demand intensifies for laws and regulations to be put in

place to protect our privacy but these regulations are struggling to keep up with the rate that data capabilities are moving.

In May of 2018 EU released a ground-breaking General Data Protection Regulation (GDPR). The GDPR has strengthen the protections around data collection. All companies must get consent for holding any person's data and they must have a record of this consent. Companies are now required to be much more transparent about all the data they hold, how the data is used, and how it is stored. Although this law is a good step forward, it replaces a law written in 1995, showing how slow these laws are to adapt.

The technology behind data collection is both impressive and somewhat meddling. During my research for this report I tried to see what data companies might have collected on me. I found an interesting page on the privacy policy section on my google account called ad personalisation. Here was listed all the information goggle knew about me that it then used to selected targeted ads. It is clear that Google has released this data to us in light of the GDPR that I have spoken about above. Most of the information was benign such as what type of movies, clothes and music I like, which can all easily be established from search history. But such details such as age, relationship status, parental status and educational status I found harrowingly accurate. They even knew that I was living in a rental property. Although google are transparent in the data they collect and what they are using it for (advertising) it still raises questions over the protection of personal data.

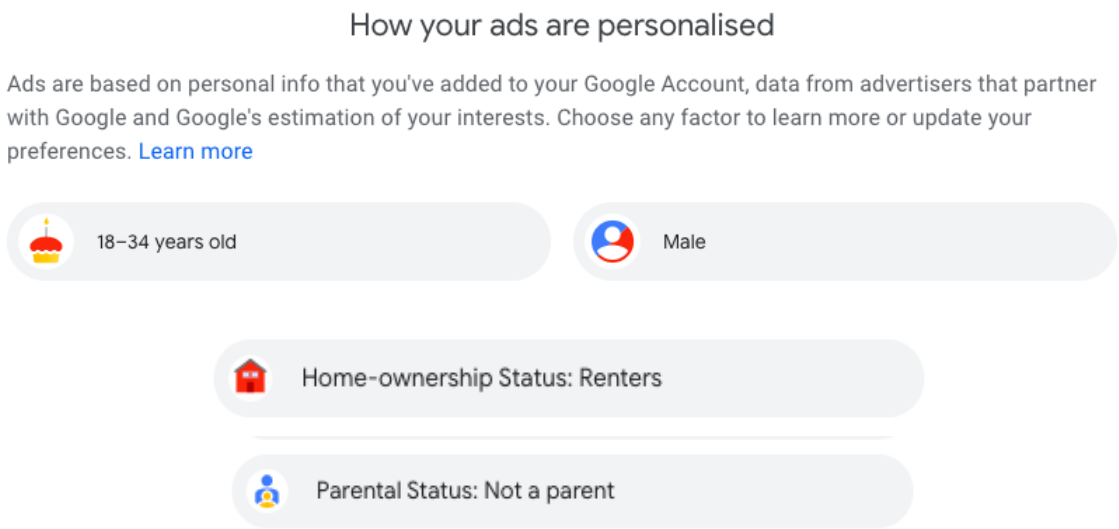

Fig.9 Advertisement personalisation from Google

Although GDPR has made strides in the right direction, there are still many concerns surrounding the protection of data. I will address some of the main ethical concerns below.

Data Collection:

The ethical concern around data collection is that there is no boundary to how much data a company can collect about their employee. It's very important for a company to find balance between gathering enough useful information while respecting the privacy of employees when measuring the work of their software engineers. I have outlined earlier how measuring certain metrics can be used to increase productivity for software engineers but increasingly companies have deviated from these metrics to try to get even more into their employees, recording things such as health records, search history and financial information. Clearly employees would be uncomfortable with people having access to this much data about their personal lives but also as previously touched on when discussing hackystat employees are uncomfortable with fine-grained data collection also as they are unaware of the extent of the data being collected. It is clear that a transparency between the employer and employee as to what data is being collected is important to set up. The new GDPR law will make a huge difference for employees in terms of transparency.

Another important issue of data collection should be that employers should restrict themselves to only collecting data about employees that is directly linked to the work they are doing in order to better access business decisions, improve employee performance and for a software developing company, improve the software engineering process.

Data Usage:

The main concern with how data is analysed is what an employer can tell about their employees based on the large amount of data that they collect. The large amount of data being collected can be used for something called predictive people analytics, which can be used as a recruitment tool to predict high performers and to predict employee likely to leave (sometimes before the employee knows) . This is where there are clear ethical questions that must be asked of companies using these techniques. Are companies entitled to know this much information? Is it fair to predetermine how successful employees might be? It also must be noted that all these data platforms and algorithms are never 100% accurate and can unfairly mislead employers.

Data sovereignty:

Data sovereignty is the concept that information which has been converted and stored in binary digital form is subject to the laws of the country in which it is located.

Many concerns arise about the enforced privacy regulations and preventing data that is stored in a foreign country from being used in court by the host country government. Widespread use of cloud computing has enhanced these issues. Employers not only have to trust their employers with their data but also trust the cloud provider is honest about where their servers and hosted and adhere to service level agreements.

5. Conclusion:

In conclusion I believe this report has shown how it can be extremely beneficial to measure a software engineer's performance. Management can track progress, assess efficiency, and use data for decision-making. Transparency is something that is crucial to making this approach effective. Managers should make it clear to their engineers what metrics are being measured the rationale behind the tracking of such metrics should be clearly identified and all of this should be communicated to software engineers. Ethical concerns must be taken into account, as it is not permissible to have question marks as to what is acceptable to measure and what is not. Until laws and regulations are fully put in place for companies, managers who wish to measure the software engineering process should do so with permission from the employees it require it from.

6. Bibliography

1. Fenton, N. E., and Martin, N. (1999) "Software metrics: successes, failures and new directions." Journal of Systems and Software 47.2 pp. 149-157.

2. A. Sillitti, A. Janes, G. Succi, and T. Vernazza, “Collecting, integrating and analyzing software metrics and personal software process data,” in Proceedings of the 29th Euromicro Conference. IEEE, 2003, pp. 336– 342.

3. J. E. Gaffney, Jr. “Metrics In Software Quality Assurance” IBM Corporation Federal Systems Division Manassas, Va.

4. Some Studies in Machine Learning Using the Game of Checkers Arthur L. Samuel

5. "Using a defined and measured Personal Software Process" by Watts S. Humphrey, published in IEEE Software, May 1996

6. Philip M. Johnson, Anne M. Disney, "The Personal Software Process: A Cautionary Case Study," IEEE Software, pp. 85-88, November/December, 1998

7. P.M. Johnson et al., “Beyond the Personal Software Process: Metrics Collection and Analysis for the Differently Disciplined,” Proc. 25th Int’l Conf. Software Eng. (ICSE 03), IEEE CS, 2003, pp. 641–646.

8. https://www.techopedia.com/definition/13296/software-engineering

9. https://en.wikipedia.org/wiki/Software_engineer

10. https://medium.com/omarelgabrys-blog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc

11. https://blogs.dnvgl.com/software/2017/10/modern-business-speed-disruption/

12. https://www.scrum-institute.org/Burndown_Chart.php s

13. https://www.agilealliance.org/glossary/lead-time/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'lead*20time))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

14. https://people.dsv.su.se/~joco2917/ft_gateway.102cfm.pdf

15. https://www.semanticscholar.org/paper/Project-Hackystat%3A-Accelerating-adoption-of-guided-Johnson/33d341507e26eb3b9b43bc99b53952d4ed3f6aa5

16. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.368.2254&rep=rep1&type=pdf

17. https://medium.com/30-days-of-machine-learning/day-3-k-nearest-neighbors-and-bias-variance-tradeoff-75f84d515bdb

18. https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26

19. http://www.learnbymarketing.com/methods/k-means-clustering/

20. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=730851