Measuring Engineering Report

Introduction

The objective of this report is to firstly scrutinise the methods of measuring the process of software engineering, secondly explore the data we consider important and relevant to our analysis and finally investigate most useful tools for visualising the collected data. I will also address the purpose of collecting and measuring data and the ethical dilemma involved. Several attempts have been made to quantify software engineering but to date there is still no perfect framework that is always a suitable solution. We will start with a simple question.

What Is software engineering?

In a paper written by Ian Sommerville, Software Engineering is defined as "An engineering discipline that is concerned with all aspects of software production". The everyday intricacies of our society has been created and maintained by Software Engineers. So much has been made available to us such as the World Wide Web. Software engineering has only been around since 1968 when it was named at a NATO conference. To keep it's growth steady it needs to be regulated, but how? In this paper I will investigate, the ways to control this discipline, how to measure and assess the data of these software engineering processes and what platforms are available to perform these processes. I will also question the best algorithmic approaches and ethics involved in this fast-growing field.

What separates software engineering from other forms of engineering is that there is no definitive answer to each problem different projects face. There are many different techniques and approaches, tools and platforms which best suit a particular projects. However in this paper, we will give an overview of a way to help these processes in general.

Software engineering is the execution of tasks that leads to the creation, delivery and maintenance of required software.

The first step is the concept. Software specification is the term for the beginning of this process where the definition of the objective of this software is, and any obstacles that may arise are identified.

The second step is the software design & implementation. Using the guidelines laid out in the first stage of development the design and programming of the software is commenced.

Next, the software goes through verification & validation. This ensuring the software complies with the customers' needs and specifications.

Lastly, the software must be continuously maintained, keeping up to date with any new customer needs or market changes. (Gabry, 2013)

Why measure data

Software engineering is a complex. It does not simply consist of writing reams of code and hoping for the best. We measure data to have a frame of reference as to how a team is performing on a project. We do this to minimise cost, to advance the production of software and perfect the delivery of the end products.

A software engineering company is focused on the best ways to produce good quality software that is efficient and tailored to the needs of the industry. To do this, a company needs to measure the performance of each contributor. They can use this data to further improve the work of these stakeholders and reduce inefficiencies. To achieve maximum efficiency each contributing body must be working to their highest capacity. In this report, we will analyse how this data can be measured and compiled. (Van Dar Voort, J) It is difficult to measure the performance of a software engineer, unlike a sales rep where the figures are easy to understand it is much harder to place value on a line of code.

When collecting data, it is crucial to have a particular goal in mind. Too often, companies spend extensive amounts in data collection software without any way to determine if the investment has yielded results. It is necessary to ask the question of what we are trying to calculate. A great software developer should have several key attributes.

Positivity

A good developer thinks about the end-product and the consumer. They are a beacon of consistent positivity and ensure the work is done to the best of their ability. They are willing to go the extra mile and use everyone’s strengths to their advantage. Because of the positive atmosphere that radiates from them, colleagues work better with them.

Communication

There is a direct correlation between good communication skills and excellent development. Excellent communicators can ask the right questions at the right time ensuring correct product specifications. A great developer can also relay complex information back to other engineers in a coherent manner, excelling at articulating complex ideas.

Time Management

Excellent time management results in a highly reliable engineer. they have a strong work ethic which involves turning up to work on time and striving to meet their workload efficiently. Balance is crucial, outstanding developers succeed in handling their clients and colleagues.

Self-improvement

Engineers with the ability, willingness and drive to constantly evaluate their work and search for self-improvement opportunities tend to outperform their peers.

Data Measurement

As with any other industry, a manager of a software engineering project needs to know how the team is performing and how the project is developing. To maximise the efficiency of a software project, there are many factors you must look at to get a good understanding of performance. I will break this analysis into two areas.

- The Engineer
- Their code

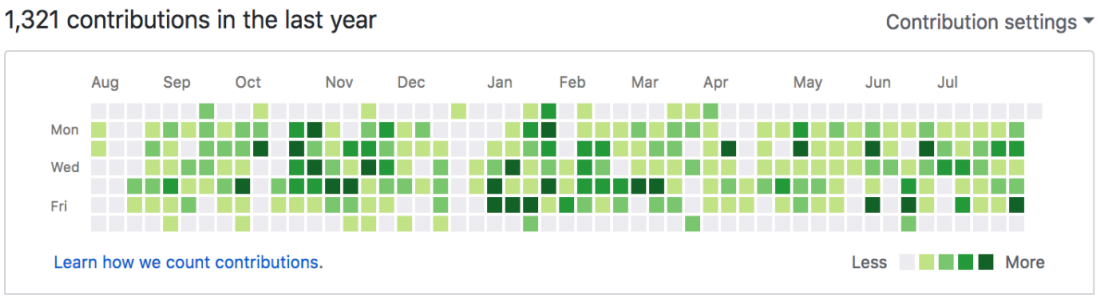
Analysing the Engineer

To understand the code being produced and how to maximise its efficiency the manager must first analyse the way the programmer works.

Number of Commits

This is an easy process to track progress. Commits are an indication of activity and are an easy way to track progress. Each time the code is changed is it highlighted. The Git

Repository logs the frequency and size of each commit and builds a report of how active a programmer is. It can indicate the time spent on a project. Here we can see when the project is most active and when it is being finished off. (Dario, J.)



Above is an example of a healthy project. The project starts strong at the beginning of December with high activity, setting the foundation, starting with the easier bits of code. Then as the code is being fine-tuned and tested, fewer commits are being done. It is not translucent, some intuition is needed to analyse a programmer through their number of commits. A programmer doing fewer commits may be working on a more complex piece of code. Someone who commits often to their repository shows that they are open with their work, they are willing to work as part of a team, letting other people see their code, give feedback or even fix it. A proud programmer that would refuse help could sometimes be damaging to a project. An engineer changing one piece of code too much might indicate a perfectionist and therefore wasting the time of a project. Hence, the number of commits is a good measure of an engineer and how they work both individually and as part of a team. (ELGABRY, O)

Time stamps

Another advantage to commits is the time stamp on them. This allows us to see when there commits are being done. It shows the speed at which a developer is working. If they like to work fast making regular minor changes or if they take their time writing long pieces of code then commits it all at once. The developer who works fast can see easily where a piece of code went wrong and revert to the point it breaks and correct it, whereas the developer who works slower with larger chunks of code is more comprehensible when reading back through their history.

Analysing the code

The quality of a piece of code is something which is more difficult to measure.

Source Lines of code

SLOC is a frequently used measure of project progress. We can use this to analyse the productivity of a project. It displays the size and complexity of the code but is not the best way of measuring a project. The first type of SLOC is physical SLOC, where the number of lines of code is counted. This is very misleading as quantity does not always mean quality. A shorting piece of code achieving the same result as a long one is always preferred. An easily comprehendible code with little repetition is ideal. The second type is logical SLOC which is slightly more difficult to measure as it is the number of statements in a piece of code. This method should be paired with other methods because on its own it encourages confusing, repetitive and messy code which is not beneficial to the project . It is important that the developer is active only if that activity is going towards tidy, pithy code. (Bhatt, K)

Test Coverage

Test coverage is a means of measuring how effectively the unit test cases test the code written by the developer. It will display all the parts of the program not visited by the test cases and which branches of conditional statements that have not been taken. Helps make sure the whole code is being run and tested effectively. (Guru99.com)

It finds areas of code untested and gives feedback on the percentage of code that has been tested. It is quick, easy, cheap and also shows useless tests that do not increase coverage. Test coverage is a valuable tool to ensure minor defects in code are checked before it reaches the market.

```
public UserControl ControlWindow
{
    get
    {
        if (_debtFetcherView == null)
        {
            _debtFetcherView = new DebtFetcherMonitorView();
        }

        return _debtFetcherView;
    }
}

public string GetMonitorName
{
    get { return _monitorName; }
}

public void UpdateView(XmlNode monitorData)
{
    if (monitorData == null)
    {
        throw new ArgumentNullException("monitorData");
    }

    _monitorData = monitorData;
    ExtractApsRunData();
    UpdateViewControl();
}
```

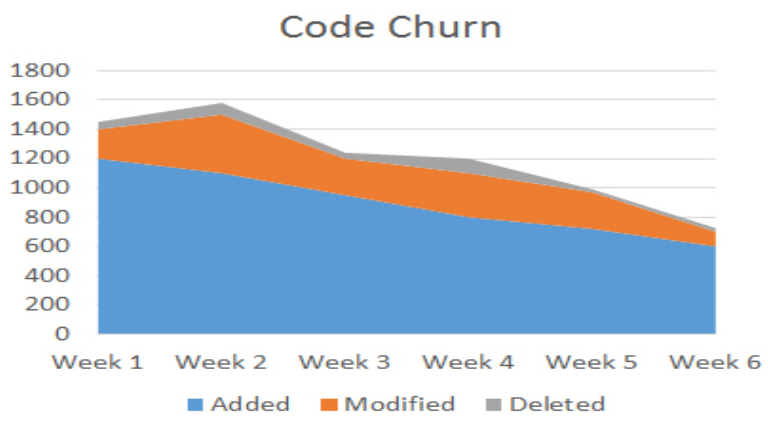
Here we can see the code in red has not been tested. If the "_debtFetcherView" is null then this condition is not tested. If there is a problem with this line the testing would not have picked it up. Therefore there is no way of knowing if this code works. The tests may

come back working but not all the code has been tested. It is important to know this before the code reaches the market. (Haunts, S)

Code Churn

Code churn is "a scope of code lines produced, changed or deleted"[9]. It shows effectively how much of the code being written by the developer ends up in the final product. You can analyse how to project changes over its life span. A high level of code churn indicated an unstable project. If the program is being frequently modified and changed it shows that they are unsure or struggling with that aspect. This is expected at the beginning of a project as the code Is taking shape but if it doesn't decrease towards the closing stages this would be a red flag for the manager. (Docs.microsoft.com)

Here is the analysis of a healthy project which starts unstable adding and changing large pieces of code but then as it progresses gets more stable and keeps its structure. (Codemanship.co.uk)



Conclusion

No one of these metrics can truly be ranked best for the analysis of a project and it's contributors. Each project varies so much between output and the variables involved in achieving that output. So it is advised to take a collection of all these tools to get a deep understanding of what is going on with the project and to recognise certain behaviours and characteristics and keep track of the projects progress.

Available platforms

Once your data has been collected it needs to be analysed. The ability to make important fast decisions concerning a project is very important. Therefore it is important for the data that has been collected to be displayed in a clear precise way. Over the years the analytics of

software metrics has grown. It is very difficult to measure the software engineering product metrics and as a result of this each project may be suited to a different platform and each company may choose different tools.

There is a magnitude of platforms available to display and measure the data we spoke of above, some are free and some are paid for. In a paper written by Johnson et al titled "Beyond the Personal Software Process: Metrics Collection and Analysis for the Differently Disciplined", we are told there are three generations in data collection. (Johnson, P)

Characteristic	**Generation 1** (manual PSP)	**Generation 2** (Leap, PSP Studio, PSP Dashboard)	**Generation 3** (Hackystat)
Collection overhead	High	Medium	None
Analysis overhead	High	Low	None
Context switching	Yes	Yes	No
Metrics changes	Simple	Software edits	Tool dependent
Adoption barriers	Overhead, Context-switching	Context-switching	Privacy, Sensor availability

PSP

The first generation of data collection started in the 1990s was the PSP method. The person software process was an intensely manually based process. The developer had to log effort size and defect information. It was also recommended they keep a stopwatch at their desk to report all interruptions. This process came with a heavy overhead in the collection of data and was not very popular amongst developers. (Johnson, P)

Leap

The next step was a more automated expansion on the PSP method. This tool helped considerably enhance project planning and quality assurance. Still, this tool was not widely adopted. While "some adoption" progresses from "no adoption" it is not the solution. Many regarded it as interruptive to their work having to stop and log everything they did. (Johnson, P)

Hackystat

In 2001 the Hackystat project begun. This eradicated the previous problems with PSP and Leap. It was a background application which collected it's needed data on it's own, without the effort of the developer. This meant the overheads are eliminated and the interruption of

switching back and forth to log your work was no longer an issue. Unfortunately, a different issue arose. Since the data is being collected in the background in such high volume, the developers don't know what data of theirs is being collected and analysed. (Johnson, P)

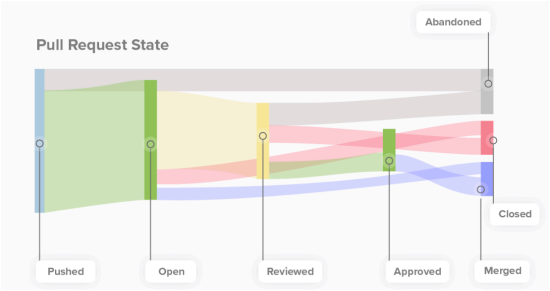
In the past few years, there have been many exciting developments in product metrics and the platforms available to analyse them. Other than the ones I have discussed there are a few other popular premium platforms available that have become very popular in recent years, the most popular being Code Climate, Codebeat, Github API, Code, Hubstaff and many more. It is as popular as it is expensive. I will now discuss the most exciting of these new platforms.

Code Climate

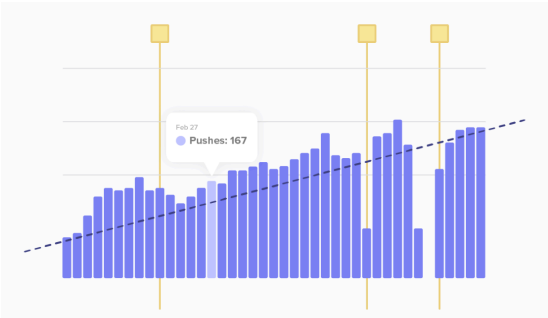
Code climate is currently the most popular and expensive service available. It is a privately owned company that enhances the developer's code quality. The Code Climate website gives a thorough breakdown of what it does for the engineer. It analyses the code in such a way that pushes coders into new levels of efficiency, laying out the weak points in their code. (Codeclimate.com)

Clear display from top to bottom of the code:

Anticipates barriers. Learning how many Pull Requests move between every possible state, then making a diagnosis to why some Pull Requests are closed or abandoned.

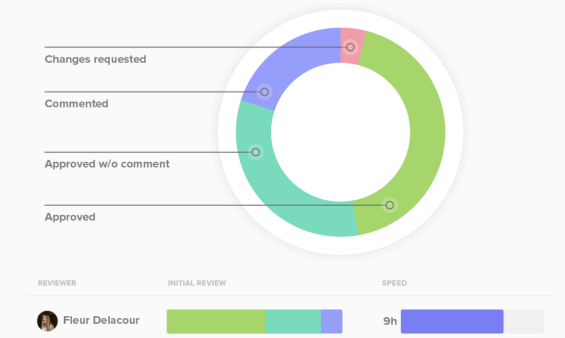


Monitors the pace of the project and your team. Gives you an unbiased answer as to whether processes changed has a positive or negative effect on the productivity of your software engineering project. The productivity is either trending up or down. (Codeclimate.com)

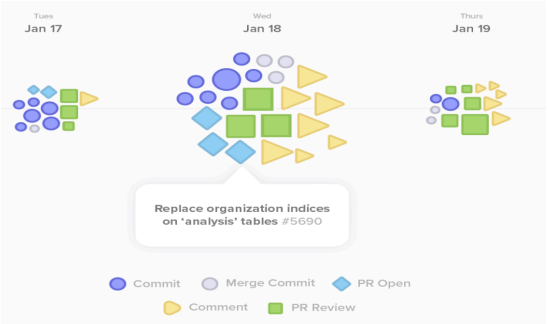


Informative information on the aspects of the projects receiving most attention:

Hits the precise balance between speed, thoroughness and code review. Sees what causes the code review process to slow down by looking at particularly long review cycles and how feedback is being done.

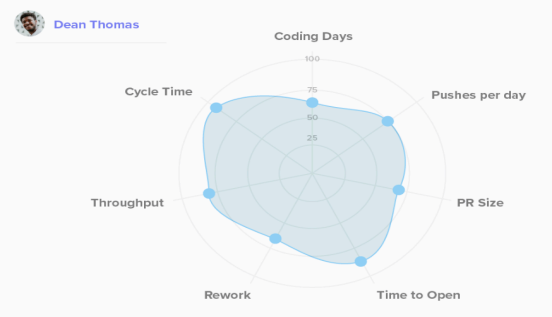


Checks the distribution of work and who is working on what. Makes sure the workload is in line with your expectations and that no member of the team is being strained. (Codeclimate.com)



Flexibility to concentrate on the unique challenges of your team:

Dissects the data based on tailor made cohorts. Although all developers have different styles, no one exists in a vacuum. Understanding how metrics are contrasted by group, team, or function in order to better identify targets of growth and expectation of baseline.

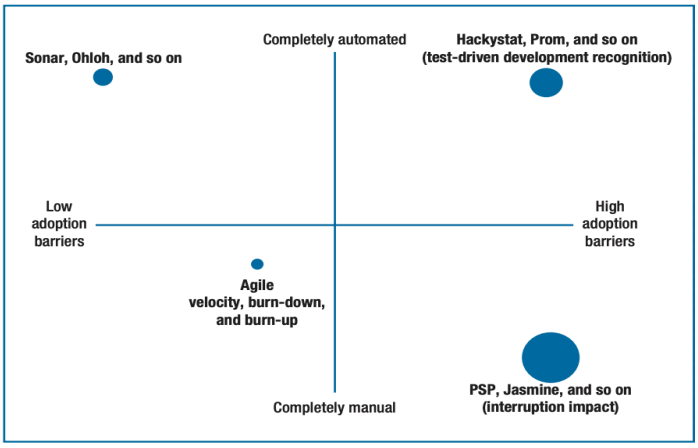


Explores deeper into your data. The flexibility of Code Climate allows you to find answers to questions that go beyond what's "out of the box"

Creates a customer report that covers all aspects of a new initiative, such as extending recruitment to members of the remote team or restructuring. (Codeclimate.com)

Other platforms

Many specialist firms in the field of computer analysis have developed platforms that are customised for particular tasks and projects, over the past few years as the importance of measuring the development process is increasing. There are a variety of other companies offering similar product apart from the ones I have discussed. Each platform having a difference in balance of Automation and adoption barriers. (Johnson, P)



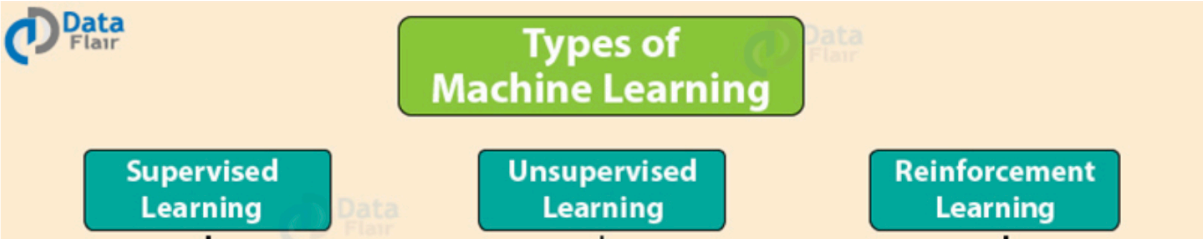
Algorithmic approaches

The above-mentioned computing platforms have numerous algorithms working in the background. The data is first collected then displayed in a way which is beneficial to its user.

It can be difficult to manually identify flaws in a program, especially in large-scale projects. A more effective approach is to analyse the data continuously through the use of algorithms. It can give us a deeper understanding of the raw data and identify any underlying mechanisms or common themes. Predictive analysis uses statistical methods to analyse data and to predict trends that may occur in the future. There are several algorithmic approaches available but since machine learning is so popular I will focus mainly on these techniques. (DataFlair)

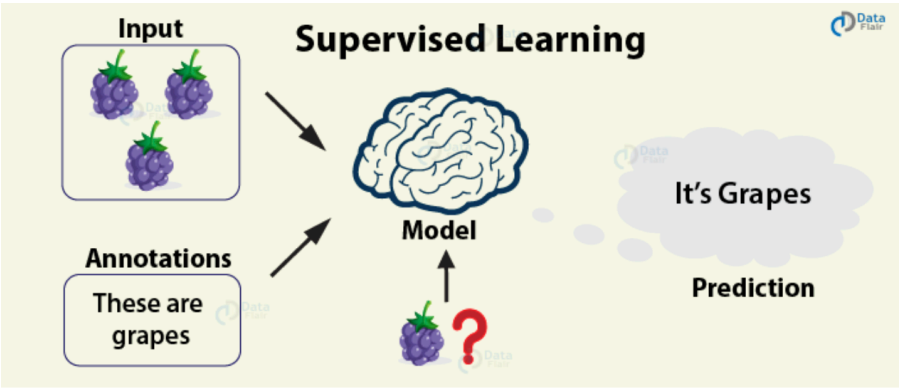
What is machine learning?

Machine learning allows systems to make autonomous decisions without external support. These decisions are made when the machine can learn from the data and understand its underlying patterns. Through the matching of patterns and further analysis, the machine returns a classification or prediction. In general, you can divide machine learning into three subsections; supervised, unsupervised and reinforced learning.



Supervised Learning

Take for example a teacher teaching a class. The teacher has the answers but the cycle of learning doesn't end until the student also knows the answer. In essence, this is supervised learning. Here, the algorithm is the student who learns from a data set provided by the teacher. The algorithm makes predictions that the instructor is correcting. The learning process continues until the set performance level is reached. (GeeksforGeeks)



Linear Regression

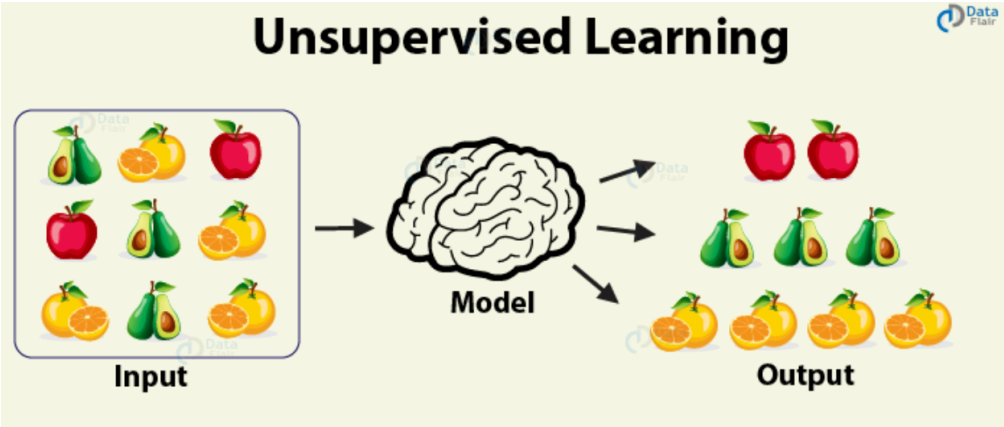
In linear regression, we test the relationship between two or more variables. We make predictions that follow this linear pattern based on these relationships. (DataFlair)

Random Forests

Random forests are an ensemble learning method for performing classification, regression and other tasks by building decisions trees and delivering output as a class that is a mean or mode of the individual trees that is underlying it. (DataFlair)

Unsupervised Learning

With unsupervised learning there is no teacher supervision, therefore the algorithm has to teach itself. The algorithm is left unattended to identify underlying patterns or structures based on density, similar segments or other similar characteristics to learn more about the data itself. The algorithm is left unsupervised to find the underlying structure in the data to learn more and more about the data itself. (GeeksforGeeks)



Clustering

Often referred to as cluster analysis is the process of grouping similar objects together that is distinct from other groups. (DataFlair)

K-Means Clustering

The objective of the k-means clustering is to divide all the observations of the data set into k clusters, each observation belonging to the cluster with the nearest mean. (DataFlair)

Reinforced Learning

This is similar to real-life where the student is released in the real world. The algorithms learn optimal actions based on trial and error. They interact with their environment and predict the ideal behaviour for the specific context. Through reward feedback, the algorithm can learn from previous decisions and improve in the long run.

Conclusion

Such algorithms' capabilities are rapidly increasing. The learning algorithms aim to minimise the error regarding the inputs received. The reliability of the decisions made by these algorithms will increase as time goes on and more inputs are fed into the model. They have yet to master the human touch so I would recommend using them in partnership with your judgment when analyzing the performance of a project. Hopefully one day these techniques will improve to the point where human intervention is not needed but we are not there yet.

Ethics

Ethics is a serious issue in today's society. Frameworks such as patient-doctor and client-lawyer confidentiality have been around for years. Protecting peoples sensitive personal information concerning online data, some murky issues have arisen in recent years questioning the protection we have online. Some of the most influential people in software engineering have pushed the boundaries and blurred the lines in relations to data collection and usage in the attempt to better their business. Facebook often changes its policies, like a kid changing the rules to a game of his own making, a subtle update when it's convenient, a knee jerk turn when its backed into a corner. Companies such as Volkswagen and Cambridge Analytica have been exposed for abusing the rules surrounding our sensitive data.

Volkswagen

In 2015 the VW engineers programmed software in their diesel cars to detect emissions tests. The vehicles would change their engine operations during the test to comply with legal emission standards. But the cars released emissions into the air under normal driving conditions that were forty times the legal limit. (Hotten, R.)

Cambridge Analytica

In a report made by the New York Times to influence the results of the 2016 presidential election, Cambridge Analytica misused personal data on more than 50 million Facebook users. The following is a breakdown of what happened made by 'InTouch solutions'.(IntouchSolutions)

1. In return for $1 – $2, roughly 270,000 people downloaded a Facebook app called "thisisyourdigitallife."
2. With users' consent, the app pulled information from their and their friends' Facebook profiles.
3. The researchers who created the app then passed the data to CA without users' permission — a violation of Facebook policy.
4. The data was then allegedly used by CA to build psychographic profiles of users and their friends, to help the Trump campaign identify voters for targeting, to provide advice on where to focus campaign efforts, and even what to say in speeches.

Data Collection

The collection of data has several legal concerns. These result from the unobtrusive collection of data. In terms of Hackysts, as I mentioned earlier, developers are often unhappy with their data being collected without their knowledge. Similar to when Facebook put in fine print written in their terms and conditions for a quiz that by partaking in the quiz you give Facebook permission to access not only their information but the information of all your friends. Managers analysing software developers productivity should be clear in defining the data they are collecting and request permission for that specifically.

Data Usage

How data is used and analysed is surrounded by several ethics. Most of this relates to keeping the analysis fair and impartial. The company must promote good values for their project, for example rewarding people for writing more lines of code will encouraging them to write poor quality code just to reach a quota. Similarly with Volkswagen, potentially too much pressure was put on their software engineers or they were encouraged to create a program that deceived the emissions tests.

Legal Aspects

On 25th of May 2018, a new European Union-wide framework known as General Data Protection Regulation (GDPR) was introduced. An accompanying directive set standards in the field of criminal offences and penalties for data protection. It is said in the GDPR that if a body is found processing data that they do not have sufficient customer consent that they can be fined up to 4% of annual global turnover or €20 million whichever is greater. As a result of this regulation, any company that seeks to measure its software engineering process must do so in a manner that is consistent with current regulation. (Citizensinformation.ie)

Transparency

Transparency in what is obtained, how it is used, and if it is passed on to any third parties is extremely important. Software engineers should always be aware of the metrics that are being evaluated by management and their reason for doing so. This will help dissolve the negative stigma of data collection amongst workers, show its utility and mitigate concerns about any abuse of personal information. The partnership between management and software engineers can be broken without accountability. There are so many platforms now to voice one's dissatisfaction or upset with problematic work conditions and this negative image could be very troublesome to a company's future. This is highly regarded amongst developers and the most skilled worker will be attracted to a company with a strong image and positive values.

Conclusion

Appealing to the human side of the developer is also very important and something that is harder to encourage through computational metrics. Reward developers empathy, conscientiousness their ability to communicate and be involved with a team. Their open-mindedness, creativity patience are more things that help a project run smoothly from start to finish and avoid conflict. A manager needs to be able to spot these attributes and encourage them within the organisation.

There is no easy way to gather and analyse data, also, there is no right or perfect way to visualise this data. Opinions toward the best approach will vary hugely between project and organisation. The evaluation of software engineering must be mutually beneficial for the management and its individuals. If an organisation is acting in an oppressive manner this may discourage a developers creativity, on the other had a lack of supervision may lead the project

astray. It is important to have a good understanding of all available tools and platforms so the management team is able to adapt to each project and its barriers.

The last point ill make in this report is the importance of having an ethical and moral stand on collecting data. With the introduction of new laws respecting privacy is no longer just proof of character but a punishable crime. I strongly advice management to be very cautious of this and be aware of any regulations and law surrounding data analysis.

References

A. Kitchenham, B. (2019). *Modeling software measurement data - IEEE Journals & Magazine*. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=950316&tag=1 [Accessed 28 Oct. 2019].

Bhatt, K. (2019). [online] Available at: https://www.researchgate.net/publication/281840565_Analysis_Of_Source_Lines_Of_CodeSLOC_Metric [Accessed 29 Oct. 2019].

Dario, J. (2019). *Measuring Developer Productivity*. [online] Hackernoon.com. Available at: https://hackernoon.com/measure-a-developers-impact-e2e18593ac79 [Accessed 28 Oct. 2019].

Docs.microsoft.com. (2019). *Code Churn Excel Report - TFS*. [online] Available at: https://docs.microsoft.com/en-us/azure/devops/report/excel/code-churn-excel-report?view=tfs-2017 [Accessed 30 Oct. 2019].

ELGABRY, O. (2019). *Software Engineering — Software Process and Software Process Models (Part 2)*. [online] Medium. Available at: https://medium.com/omarelgabrys-blog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc [Accessed 26 Oct. 2019].

Guru99.com. (2019). *Test Coverage in Software Testing*. [online] Available at: https://www.guru99.com/test-coverage-in-software-testing.html [Accessed 28 Oct. 2019].

Haunts, S. (2019). *Unit Test Coverage, Code Metrics, and Static Code Analysis*. [online] Stephen Haunts { Freelance Trainer and Writer }. Available at: https://stephenhaunts.com/2013/02/18/unit-test-coverage-code-metrics-and-static-code-analysis/ [Accessed 7 Nov. 2019].

Kravchenko, I. (2019). *7 Metrics To Measure Software Quality in The Most Efficient Way*. [online] Diceus. Available at: https://diceus.com/top-7-software-quality-metrics-matter/ [Accessed 28 Oct. 2019].

Purna Sudhakar, G. (2019). [online] Available at: https://scindeks-clanci.ceon.rs/data/pdf/1452-4864/2012/1452-48641201065S.pdf [Accessed 29 Oct. 2019].

Van Dar Voort, J. (2019). *Commits Do Not Equal Productivity*. [online] GitLab. Available at: https://about.gitlab.com/blog/2016/03/08/commits-do-not-equal-productivity/ [Accessed 7 Nov. 2019].

Citizensinformation.ie. (2019). *Overview of the General Data Protection Regulation (GDPR)*. [online] Available at: https://www.citizensinformation.ie/en/government_in_ireland/data_protection/overview_of_general_data_protection_regulation.html [Accessed 4 Nov. 2019].

Codeclimate.com. (2019). *Understand Each Step of Software Development Life Cycle | Velocity*. [online] Available at: https://codeclimate.com/velocity/understand-diagnose/ [Accessed 31 Oct. 2019].

Codemanship.co.uk. (2019). *20 Dev Metrics - 2. Cost of Changing Code - Software People Inspiring*. [online] Available at: http://codemanship.co.uk/parlezuml/blog/?postid=1433 [Accessed 30 Oct. 2019].

DataFlair (2019). *Learn Types of Machine Learning Algorithms with Ultimate Use Cases - DataFlair*. [online] DataFlair. Available at: https://data-flair.training/blogs/types-of-machine-learning-algorithms/ [Accessed 2 Nov. 2019].

GeeksforGeeks. (2019). *Top 10 Algorithms every Machine Learning Engineer should know - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/top-10-algorithms-every-machine-learning-engineer-should-know/ [Accessed 2 Nov. 2019].

Hotten, R. (2019). *Volkswagen: The scandal explained*. [online] BBC News. Available at: https://www.bbc.com/news/business-34324772 [Accessed 7 Nov. 2019].

IntouchSolutions (2019). [online] Intouchsol.com. Available at: https://www.intouchsol.com/wp-content/uploads/CambridgeAnalyticaPOV_IntouchSolutions.pdf [Accessed 3 Nov. 2019].

Johnson, P. (2019). *Beyond the Personal Software Process: Metrics collection and analysis for the differently disciplined*. [online] Available at: https://www.researchgate.net/publication/4016775_Beyond_the_Personal_Software_Process_Metrics_collection_and_analysis_for_the_differently_disciplined [Accessed 30 Oct. 2019].

Johnson, P. (2019). *Searching under the Streetlight for Useful Software Analytics - IEEE Journals & Magazine*. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6509376&tag=1 [Accessed 1 Nov. 2019].