Measuring Software Engineering

Andrew Tobin
183182784

Table of Contents

Intro

In this report we will analyse the measurement of software engineering. Before diving into detailed analysis of how it is measured, we must first ask ourselves why software engineering might be measured? Industry and society as a whole, have long been obsessed with efficiency and how to enhance performance whether this be in how cost effective a business model is or how energy efficient the lightbulbs in our own homes are. This is the fundamental idea behind measuring software engineering, identifying areas where we can improve a product or a process by looking at different measurement metrics. Another reason for measuring software is to help people better understand the process in question. Understanding the process or product also helps the users to better monitor and control it.

How do we measure software engineering?

There are many different metrics used to measure software engineering, they can measure different attributes of the software or the software development team such as the reliability of the code and how often the code fails or the productivity of the team and how often they churn out new code on a project. In this section of the report we will analyse the different measurement categories and the metrics used to measure them, we will also analyse the caveats associated with each category.

We must first define what makes a good metric. Below are five criteria for defining a good metric.

Time: Metrics should give an indication of how a team is performing over time. They should not provide a static view of how a team is performing at one instant alone, if this was the case management would not be able to determine if there has been improvement. In essence you should be able to compare a present version of a team with its past version

Actionable: Metrics should point to an area that needs improvement. They should give a clear snapshot of what action needs to happen for the company or software team to improve.

Consistent: Metrics should be consistent, this means that what they measure is scaled so that comparisons are made in the same units without major variability due to outside factors where possible.

Comparable: Metrics should be useable in comparing between different teams, whether this is internal or external. Metrics are useless if you cannot use them as a benchmark for comparison.

Business Oriented: The metric should highlight the value or lack thereof of the work carried out by the team for the business, whether this be in monetary units, time-saved or customer satisfaction (Lermusi, 2003)

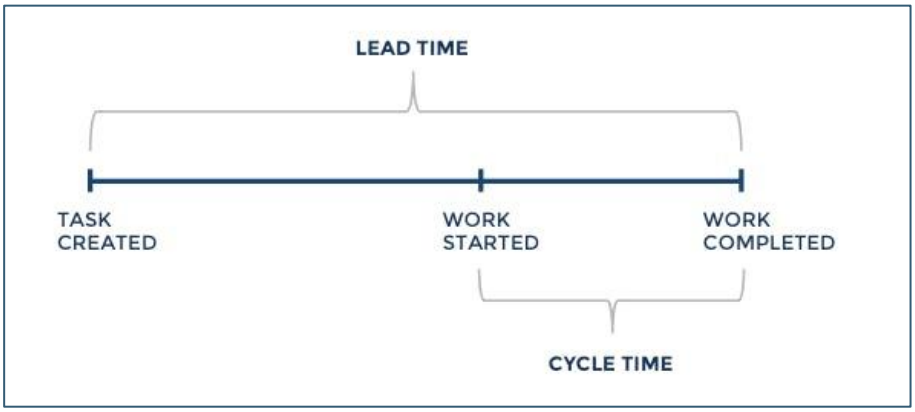
Measuring Productivity

When a software development team are developing new code for a product managers might want to analyse the teams progress and their productivity. This can help the manager to assess whether they will meet deadlines and perhaps show the manager areas that the team can improve.

Some of the different productivity metrics used are “Lead Time”, “ Cycle Time”, “ Team Velocity” and “Active Days” (Altvater, 2017). We will take a look at what these metrics measure and also raise some caveats for production metrics.

Lead Time is the amount of time it takes to complete a task after it has been the task has been set. It is important to distinguish between lead time and cycle time, the image below is a useful tool for distinguishing between the two.

Cycle time is the time between starting the task and completion of the task. It is important to measure both lead time and cycle time, often times there can be issues between when the task is set and when the software development team begin working on the task, this could be a database crashing and the task being deleted or any other number of issues. Cutting down on the lead time limits the time for potential errors such as these to occur (Bluemel, 2020). Cycle time tracks the productivity of the software development team, a lower cycle team indicates a team that is moving quickly but there are caveats to be aware of when using these metrics which we will discuss later in this section



Team velocity is very similar to cycle time, team velocity is the amount of software the team creates in a given time period called a sprint. Sprints usually last about two weeks. Using team velocity to compare software development teams is generally not advised as each software team may have different lengths for their sprints or may measure units of code differently, it is usually used as an in-house measure.

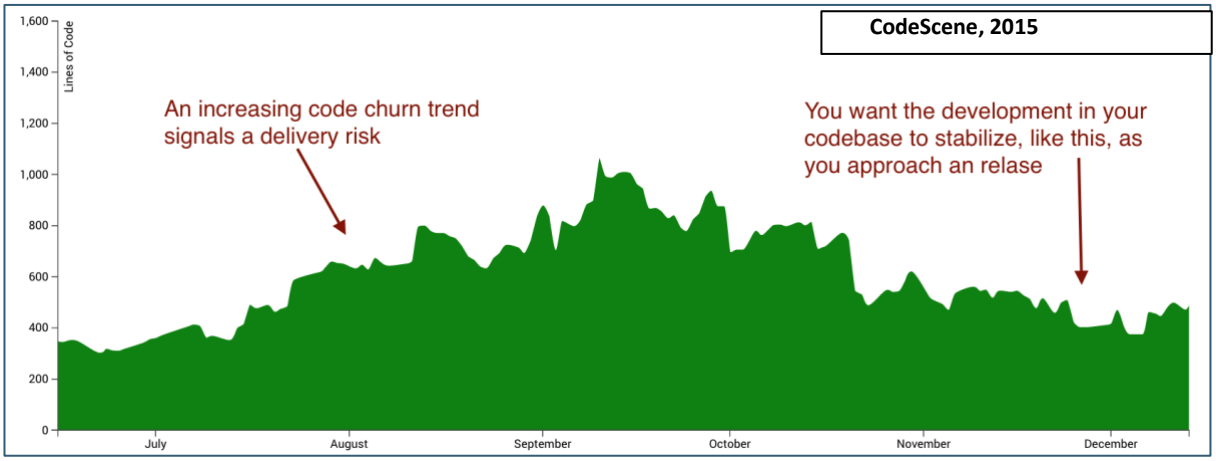
Active-days is a measure of the days a software team spends developing code and contributing it to the project. Active days don’t include days spent planning or meeting with customers to identify the requirements of the software. They can help us to see the effect of interruptions to code production such as those mentioned above (Pluralsight, 2016).

While all these metrics can provide an insightful measure of the productivity of the software development team it is critical that we understand their downsides and trapdoors. Lead time and cycle time can give us a great idea of how quickly a team can complete the project however they give us no measure of the quality of the software delivered to the customer, speed does not always equate to quality and efficiency. Likewise Active days show us the cost of the interruptions however without meeting with the customer to discuss the specifications of the software the team may develop a product that is not fit for the purpose.

Measuring Efficiency

As pointed out above speed does not equate to quality and as such productivity metrics should be used hand in hand with efficiency metrics and durability metrics which we will discuss later. Efficiency metrics measure the amount of code in the software that is useful. Efficiency metrics include “Code Churn”, “Impact”.

Code churn refers to the amount of code edited or deleted after being written. This is very common in software engineering as programmers proof and modify code to improve it. Generally low code churn indicates more effective code as it requires less editing and refinement. This isn’t always true however, code churn is a necessary part of programming, refining and enhancing code is good practice. It is when code churn begins to spike or drop excessively that should be cause for concern.



Impact measures the effect that changes to the code have on the project as a whole, impact encourages meaningful changes rather than adding lines of dead code. Impact is based on the amount of files the changes effect as well as the number of lines of code in the changes and finally on the amount of change caused by the new code. This is a double edged sword as impact can have both positive and negative effects on the project.

Measuring Security

Code can be swiftly developed and highly efficient but if it is not secure and safe to use it is very dangerous, especially for huge corporations storing sensitive data such as banks or hospitals. There are numerous metrics used to measure the security of software and these should be monitored closely to ensure the software team is ensuring the code is safe to use.

Mean time to repair (MTTR) is a measure of how long it takes the software team to repair the software after there has been a breach or crash. This is self-explanatory the shorter the MTTR the better.

Other security metrics can include the defects per lines of code or the amount of defects identified within the code over time.

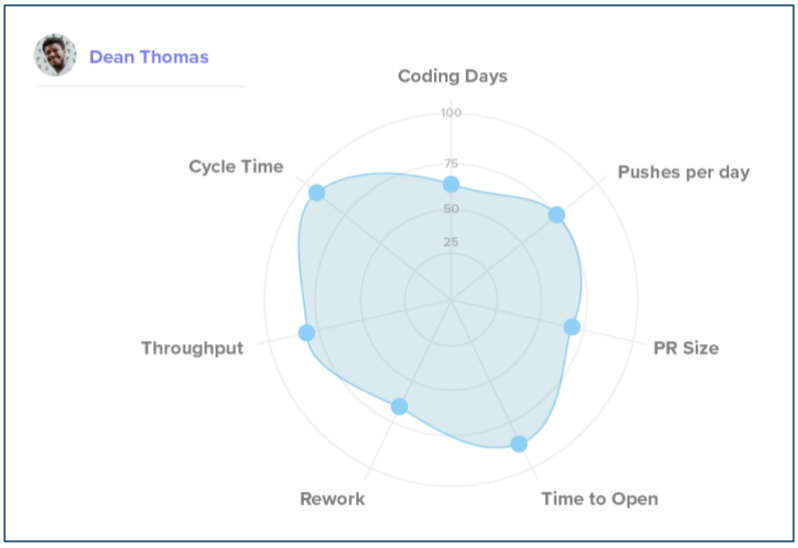
You can also assess the security across other metrics such as endpoint incidents which is the amount of viruses per successful logins, and other metrics such as number of identified risks or number of virus infections. As with MTTR the lower these are the better

Platforms for Software Measurement

As the world has become more digitised in the past 20 years data has become a commodity, with this, data analytics has become a booming industry spawning a plethora of successful analytical tools and platforms that can help ordinary open the data black box so to speak. Platforms such as Code Climate, Hackystat and PSP have helped people and organisations to make use of their data, visualising and explaining it in ways that were before now impossible. In this section we will discuss some of these platforms and assess their performance against other platforms in the industry.

Code Climate

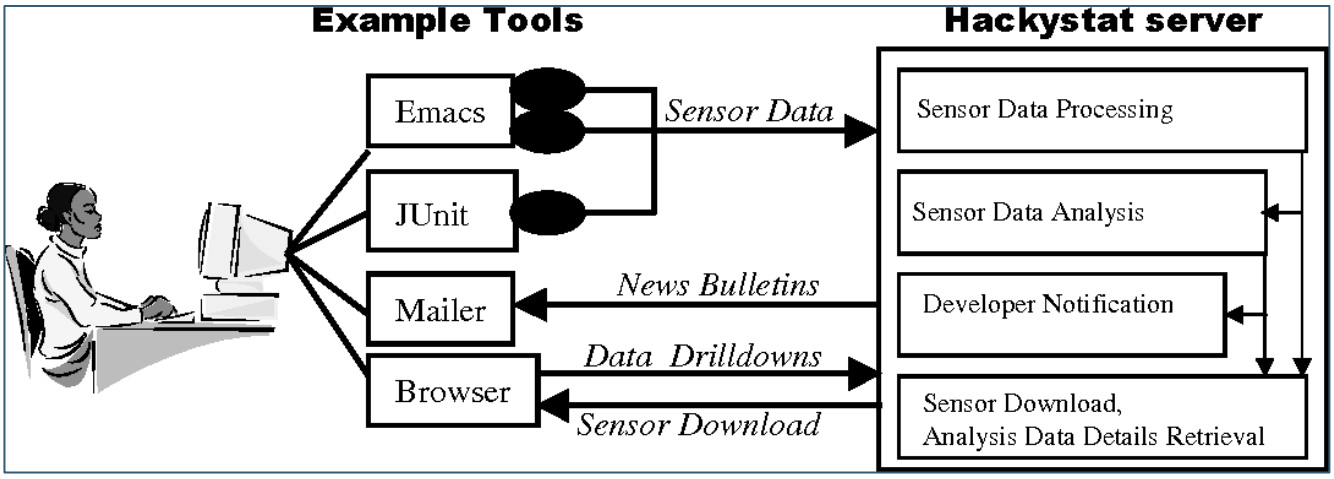
Code Climate was founded in 2011. It is compatible with Git repositories and provides simple and easy to understand metrics for assessing the quality of code that are automated limiting the possibility of human error. Code Climate provides each repository with a GPA ranging from 0 to 4.0 to assess the overall quality of the repository. It also provides measurements of the maintainability of the code and the test coverage of the code. It also offers a wide range of different analysis from seeing how pull requests are processed and where they might become stuck to analysing different members of the software team. Below we see an analysis chart of one employee generated by Code Climate that analyses their work across some of the metrics discussed in the previous section.



There are many pros to Code Climate, it has a very intuitive user interface and provides outstanding visualisations of data like the one above. It is also highly dynamic and integrates with numerous different business platforms such as Github, Slack, Campfire and HipChat.

Hackystat

"Hackystat is a software development metrics collection tool that focuses on the individual developers. Hackystat is able to provide a developer with a personal analysis of his or her unique processes" (Tomosoda & Leung, n.d.). Users install Hackystat sensors into editors, their development systems and configuration management systems. The sensors then monitor the development processes and Hackystat then sends them to the Hackystat webserver where users can access the data collected and produce reports and analysis on their development processes (Johnson & al, 2005). We see in the graphic below how the Hackystat process works.



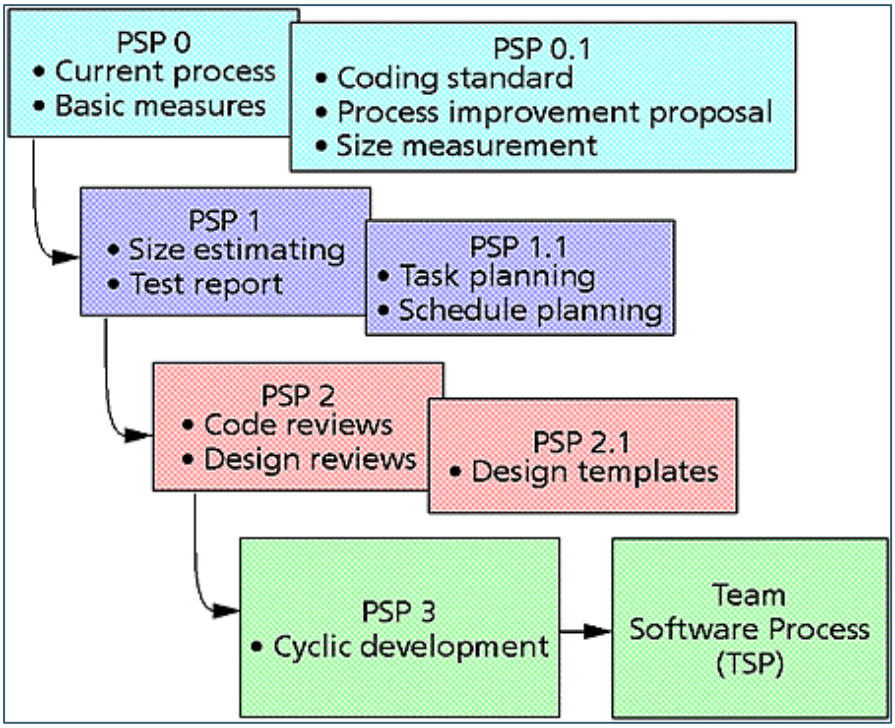
Hackystat's automated method of collecting data and analysisng it in its web server is highly accurate and reduces human error. While it is very effective in gathering and analysing data I would argue that its data visualisation capabilities are not quite at the same standard set by Code Climate. Below is a snippet of the Hackystat interface it is not very intuitive to use especially to a layman unversed in software engineering.

Project (Members)	Coverage	Complexity	Coupling	Churn	Size(LOC)	DevTime	Commit	Build	Test
DueDates-Polu (5)	63.0	1.6	6.9	835.0	3497.0	3.2	21.0	42.0	150.0
duedates-ahinahina (5)	61.0	1.5	7.9	1321.0	3252.0	25.2	59.0	194.0	274.0
duedates-akala (5)	97.0	1.4	8.2	48.0	4616.0	1.9	8.0	5.0	40.0
duedates-omaomao (5)	64.0	1.2	8.2	1568.0	5597.0	22.3	59.0	230.0	507.0
duedates-ulaula (4)	90.0	1.5	7.8	1071.0	5416.0	18.5	47.0	116.0	475.0

Hackystat also raises an ethical question. The sensors can gather data unbeknownst to the user which may violate the users privacy if they do not consent to their data being collected. We will discuss the ethics of software measurement later in this report.

Personal Software Process (PSP)

PSP was developed in 1993 by Watts S. Humphrey. PSP is a framework for software development grounded in a structured and disciplined approach to software development. The framework acts as a personal guide for software engineers and allows them to assess themselves as they work through the development process. The graphic below shows the steps involved in the framework. It is broken into four sections PSP0 to PSP3. The first section PSP0 encourages the programmer to first write their code then carry out a retrospective assessment of their work and seek for areas to improve upon. After this the programmer can move to PSP1 which involves testing and scheduling. The programmer iterates the steps and assesses their personal performance against the framework. This may seem rather primitive when compared to Code Climate and Hackystat two automated measurement platforms but PSPs ad hoc method paved the way for these platforms when it was first introduced to the world back in the 90s. It is often cited as the match that lit the flame.



Algorithmic Approaches

As we can see software measurement has moved away from human entry platforms such as PSP to more automated platforms such as Hackystat and Code Climate. In this section we will analyse the algorithms behind these automated processes, specifically the machine learning (ML) algorithms that drive more and more automated platforms these days as AI begins to become one of the cornerstones of data analytics.

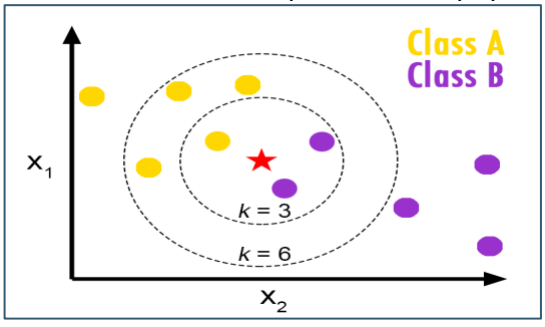
Firstly what is machine learning? Machine learning is as defined by the Oxford Language dictionary "the use and development of computer systems that are able to learn and adapt without explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data" (Oxford Languages, 2020). There are three types of machine learning: supervised learning, unsupervised learning and reinforcement learning. We will discuss both approaches and give examples of some algorithms that fall under both categories.

Supervised Learning

Supervised learning consists of an algorithm learning to construct a mapping function that maps inputs from the user to outputs. Essentially the algorithm constructs a function $Y = f(x)$ where Y is the output or dependent variable and f(x) is the function that takes inputs x from the user. It is named supervised learning precisely because the inputs are provided by the user, the user can act as a teacher for the machine helping it to learn the optimal mapping function. Supervised learning algorithms branch into two subcategories: Classification and Regression.

Classification Algorithms

Classification is where each new data point that enters the population is assigned membership of a group or class of data points that the algorithm deems similar to each other. An example of a classification is K-Nearest Neighbours. K-Nearest Neighbours assigns the new data point to the class that has the larger proportion of neighbours from a number K that are close to the new data point entering the population of data. K-Nearest Neighbours does not take into account the spread of the population data.

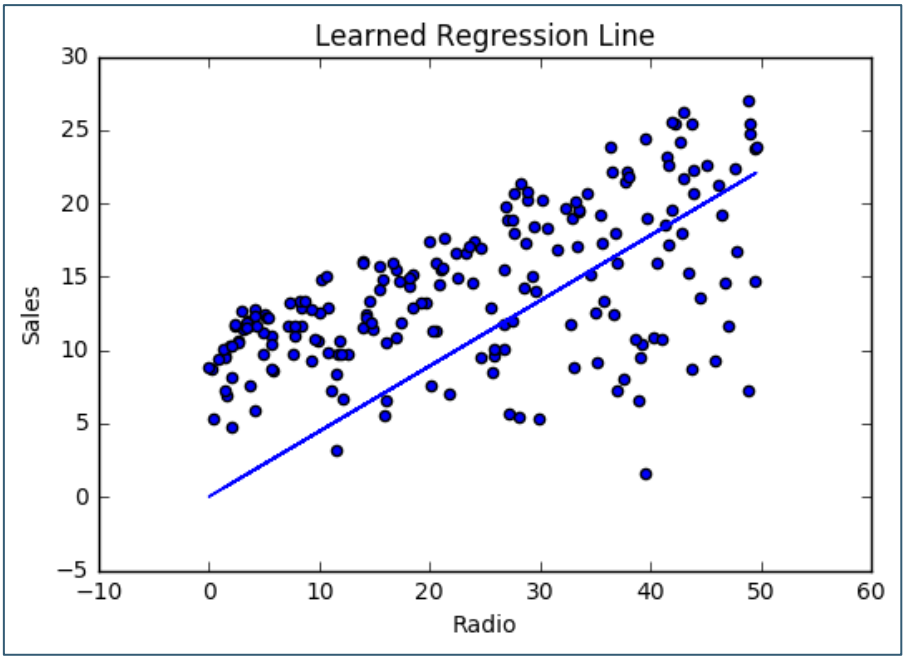


(HelloACM, 2016)

In the image above we can see that if K = 3 we would assign it to Class B as it has more data points near our new data entry, however if K were expanded to 6 this would change. As such it can be difficult to choose the optimum K that delivers accurate results. This is where the machine learning helps us, it can run through thousands of iterations to choose the optimal value for K. (HelloACM, 2016)

Regression Algorithms

We might use Classification algorithms to classify categorical data, such as deciding in a mammal or reptile based on a set of specifications such as whether they lay eggs or if they have fur etc. These classification algorithms become redundant when we want to plot or predict continuous data however such as the height of people in a course for example. For these types of datasets we would instead use a regression algorithm such as linear egression. Linear regression takes data from the user and predicts values by estimating variables such as the slope (the correlation between two or more variables) and the intercept. The machine learning algorithm can decide the optimal slope around which data fluctuates.

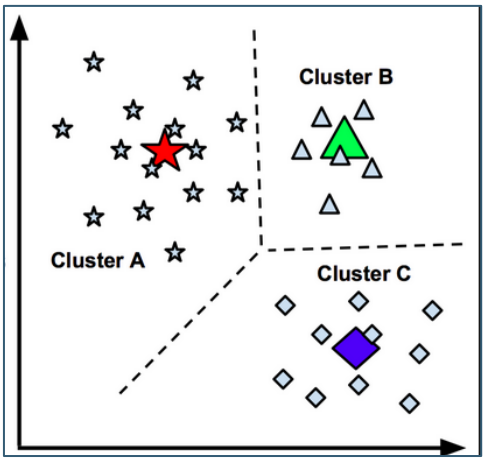


Unsupervised Learning Algorithms

In supervised learning the user supplied the data and guided the machine learning to the desired outcome in unsupervised learning the machine is given data and expected to analyse itself. The goal is for the machine to discern patterns within the data without assistance from the user. Unsupervised learning tends to fall into one category called clustering, this is where the machine identifies clusters of data points that are similar to each other and groups them together.

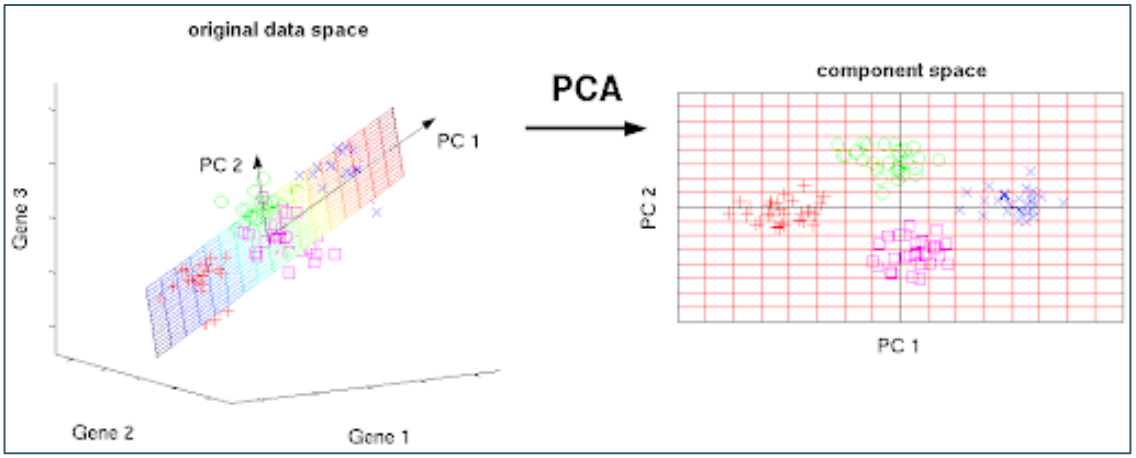
K-Means Clustering

Much like K-Nearest Neighbours K-Means Clustering uses a value K to break the data into clusters. In this instance however k is the number of clusters to split the data into. In the image below k would equal 3. K-Means Clustering works by splitting the data into k randomly selected centroids, in the image below these are the coloured shapes. The algorithm then iterates to find the optimum position of each centroid, the data tends to cluster around these centroids.
(Garbade, 2018)



Principal Component Analysis (PCA)

Principal Component Analysis is applied to datasets that contain many different variables, the idea behind it is can we form a linear combination of a smaller subset of these variables that defines the data. PCA seeks to decompose the data to a subset of the variables that are most important in defining the data, in the process it reduces the dimensionality of the data, this is done to define an internal structure within the data.
In the picture below we see how a 3D data set can be transformed to a simpler 2D one.



Reinforcement Learning

Reinforcement Learning is very like training a dog, the algorithm learns to carry out a series of decisions based on receiving a reward for a correct decision and a penalty for a wrong decision. Over time these rewards encourage the algorithm to carry out as many correct decisions as possible and minimise the amount of incorrect decisions made. You can teach the algorithm to prioritise some decisions over others by offering more rewards for certain decisions than others, this allows the algorithm to be tailored to any number of uses.

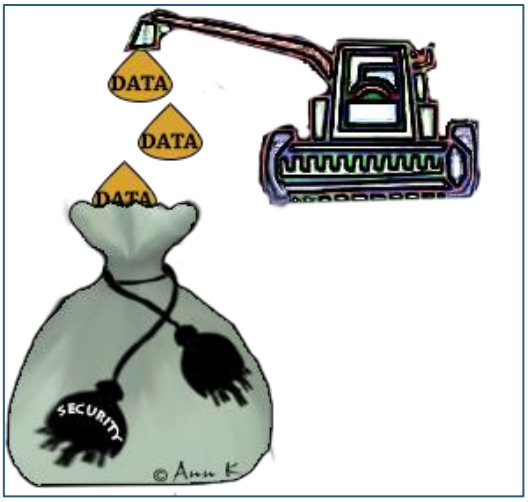
Ethical Concerns Surrounding Software Measurement

As touched upon earlier in the report, there are some ethical concerns with regards to the measurement of software. While the measurement of software can greatly increase productivity and efficiency it is important to consider whether the data has been collected, measured and interpreted ethically.

Collecting Data

Technology has become so pervasive that there are very few facets of life that are untouched by it, from Roomba hoovers to smart fridges, technology has not only proliferated the workplace but has now made its way to the home. It is also constantly gathering data and sometimes silently so, this raises numerous ethical concerns. As seen earlier with Hackystat it is important that people are aware that their data is being harvested and they are in effect being monitored, otherwise there exists a serious breach of privacy. These breaches of privacy have become more and more prevalent as technology has evolved over time.

In the workplace there are numerous examples of employers playing big brother to staff, just this year Barclays were embroiled in a scandal with regards to monitoring their employees. Using Sapience Analytics software Barclays were measuring the amount of time their employees were spending at their desks, they ceased this practice after facing backlash regarding their employees' privacy being breached (Forbes, 2020). Knowing or suspecting that you are under constant surveillance by your employer can prove stressful and seriously dent team morale in the workplace.



Another example of unobtrusive data collection outside the workplace this time, is the Roomba hoovers which have been found to map the layout of your home and could in the future sell these layouts to companies such as Amazon to increase furniture sales. It is important that the users of these devices are made aware that their data is being collected (Astor, 2017).

Transparency is one of the cornerstones of ethical practices, companies should make it explicitly clear to employees and customers that their data is being collected and monetised.

Another aspect of data collection that can raise ethical concerns is when data is collected with bias. Data can heavily influence key decisions and as such should be collected objectively without bias to ensure there is no decision bias.

Data Usage and Interpretation

Companies also have an obligation to use and interpret data ethically. Data can be used to influence major decisions it is important that these decisions are ethically sound and not nefarious or corrupt. Data is also easily manipulated and needs to be interpreted correctly, consistently and fairly.

Data has become more and more powerful as a medium for influencing decisions, we have seen numerous scandals such as Cambridge Analytica's use of Facebook users data to influence the 2016 election. It is imperative that data is collected with virtuous intent for its use.

It is also important that data is interpreted correctly, metrics can give us great insight into snippets of larger problems but it is very important that we consider any outside factors that might influence the data that is not accounted for by the metrics. For example measuring employees productivity by a single metric "sales figures" can take the human aspect out of a job evaluation, some employees may contribute intangible value to the company that is not displayed in the data. It is important to consider all factors when interpreting data.

It is also vital that we interpret the data consistently and fairly. Data can remove the human aspect from the one side of the equation, however we need to remove human bias from the other side when interpreting the data. If the data points to your favourite employee under-performing we must remove emotional thinking and act on the data the same as any other employee for example.

Conclusion

In conclusion, this report has reviewed how data is measured, what makes a good measurement metric and their limitations. We analysed the different platforms we can use to collect, measure, analyse and visualise data and compared them to one another. The algorithmic approaches to measuring and analysing data, specifically the machine learning techniques, were explored, and finally we examined the ethics surrounding the collecting, using and interpreting data. Software has become a central part of society as we plough through the 21st century, it is important that we have a strong understanding of the back-end of the software that is driving innovation and productivity forward. It is equally important to understand its limitations and downfalls, so that we can fill in the gaps with a human touch.

Bibliography

Altvater, A., 2017. *Stackify.* [Online]
Available at: https://stackify.com/track-software-metrics/
[Accessed 22 Novemeber 2020].

Astor, M., 2017. Your Roomba May Be Mapping Your Home, Collecting Data That Could Be Shared. *The New York Times*, 25 July.

Bluemel, A. D., 2020. *Clubhouse.io.* [Online]
Available at: https://clubhouse.io/blog/lead-time-what-is-it-and-why-should-you-care/
[Accessed 22 November 2020].

CodeScene, 2015. *CodeScene: Code Churn.* [Online]
Available at: https://codescene.io/docs/guides/technical/code-churn.html
[Accessed 23 November 2020].

Forbes, 2020. Big British Bank Barclays Accused Of Spying On Employees—This May Be The New Trend. *Forbes*, 13 August.

Garbade, M. J., 2018. *Towards Data Science.* [Online]
Available at: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1#:~:text=5%20min%20read-,K%2Dmeans%20clustering%20is%20one%20of%20the%20simplest,popular%20unsupervised%20machine%20learning%20algorithms.&text=In%20other%20wo
[Accessed 24 November 2020].

HelloACM, 2016. *HelloACM.* [Online]
Available at: https://helloacm.com/a-short-introduction-to-k-nearest-neighbors-algorithm/
[Accessed 24 November 2020].

Johnson, P. M. & al, e., 2005. "Improving software development management through software project telemetry.". *IEEE software,* Volume 22.4, pp. 76-85..

Lermusi, Y., 2003. *ERE: Recruiting Intelligence.* [Online]
Available at: https://www.ere.net/characteristics-of-a-good-metric/
[Accessed 23 November 2020].

Oxford Languages, 2020. *Oxford Languages.* [Online]
Available at: https://languages.oup.com/
[Accessed 24 November 2020].

Pluralsight, 2016. *Pluralsight: 5 DEVELOPER METRICS EVERY SOFTWARE MANAGER SHOULD CARE ABOUT.* [Online]
Available at: https://www.pluralsight.com/blog/teams/5-developer-metrics-every-software-manager-should-care-about
[Accessed 22 November 2020].

Tomosoda, C. & Leung, B., n.d. *Configuration Management and Hackystat: Initial Steps to Relating Organizational and Individual Development,* Honolulu: Department of Information and Computer Sciences University of Hawai'i.