# Measuring Engineering

## Introduction

It is inarguable that software engineering has developed tremendously over the past number of decades, and is continuously advancing and branching into new unexplored territory. As our exploration of software engineering as a discipline continues, we must develop certain standards for coding, and identify the major differences between good and bad coding. However, separating good and bad engineers boils down to more than measurable data, which could include measurements of time on projects, the quality of the code/tests produced and the size of the project. It is my opinion that the software engineering process is more efficiently measured through softer aspects, rather than easily measurable data and figures. For an engineer to truly excel they must have a self-direction, autonomy and gain a sense of purpose from their work. Therefore, it would seem reasonable to measure job satisfaction in conjunction with time, quality and size aspects.

Businesses who wish to measure the software engineering process have many resources available to them. Firstly, businesses can develop analytics processes themselves, but this can be costly and complicated. Secondly, businesses can choose to "borrow" analytics processes from other companies, specifically Parasoft or Code Climate, who can assess the software engineering process and report this data back to the user at a reasonable cost. Finally, businesses can choose to implement a hybrid approach whereby they use multiple companies to collect the exact data they are looking for. Whichever method is selected it is important to remember that the mathematics cannot explain human behavior, and should be used in conjunction with logic and reasoning.

These companies will use algorithms that focus on data reduction and data clustering, and will implement artificial intelligence to do so. These companies take data sorting algorithms, and train this code to make decisions- in our case separating good engineering practices from poor ones. Techniques implemented include Principal Components and Clustering Analysis, which I will discuss in further detail. Although some might find this analysis of an engineer's performance invasive, it is necessary to see where we can make improvements in a fast paced, every-growing industry. I maintain that so long as EU Regulations are respected, it is at the CEO's discretion what data he/she wants to gather on workers, and how he/she plans on gathering it. When used in an appropriate manner this will greatly improve the work of engineer's and drive them towards new improvements.

## Measurable Data

Every day we create 2,500,000,000,000,000,000 Bytes of data. This is enough data to fill 10 million Blu-ray discs, the height of which stacked, would measure the height of four Eiffel towers on top of one another! However, as 90% of this data was created in the last two years, we are still trying to assess its relevance, and how it can be used for our benefit.

When examining a software engineer's productivity, there are three main branches of analytics- time, size and quality. It is not just what we measure that matters, but how exactly we plan on measuring it. We cannot say one engineer is better than another because he completed a project faster, if we do not know the size of the respective projects. We cannot say one engineer is better because he has longer code, if we do not know how much of the code was reused or what the base code was. Examining quality seems to be a favorable method of assessing performance, however this too poses challenges.

In this section I will explore the relevant data we can use to measure and assess the software engineering process, focusing on not both hard numbers, such as time frames and the number of tests implemented, and softer elements including job satisfaction and communication. I find the secret behind every successful and valuable software engineer is autonomy, mastery and purpose. But the real question is, how can we measure this?

## Types of Data

1. *Time*

   When measuring productivity, it is vital to consider the amount of time spent on each process of a project. By setting attainable deadlines for each stage of growth, a sense of determination and achievement can be established among coworkers, increasing productivity. However, failing to meet such targets could have a contrasting affect.

   As discussed by Humphrey in "The Personal Software Process", we can see that this calculation depends on three factors- the time work started, the time work finished, and any interruption time. Events such as calls, breaks or someone interrupting to ask questions detracts from time spent on a project. Considering interruption time is completely random, omitting it from our calculation would cause major random error in all time calculations and hence reduce accuracy.

   Lowe builds on the importance of time measurement in his article, "9 metrics that can make a difference to today's software development teams", highlighting the importance of planning and process improvement and further splitting time into four separate components. The measurable data he includes is summarized the table below:

| Measure | Description | Quality implications |
|---|---|---|
| Lead time | The time it takes to go from an idea to delivered software. Work to simplify decision making and reduce wait time. (Leadtime includes cycle time) | If there is little lead time on a project, the design of the project will be unsatisfactory. If engineers produce the design whilst coding it will be undocumented and most probably poor. |
| Cycle time | How long does it take to make changes to software systems and deliver those changes into production? | Software is an ever-evolving industry. Good quality code is easily altered to cope with this. |
| Team velocity | How many "units" of software the team typically completes in an iteration? This should not be treated as a success measure, but to plan iterations. | If too much code is written too quickly it will have many defects. Velocity shouldn't be measured in lines of code but is a measure of work rate and concentration. |
| Open/close rates | How many production issues are reported and closed within a period? | Rate of extension or modification of code. |

2. *Size*

   *"Measuring programming progress by lines of code is like measuring aircraft building progress by weight."- Bill Gates*

   Say a painter was interested in computing the amount of time he would spend painting a wall. His first step in predicting this would be to measure the surface area of the wall. The same methodology is applied when computing how much time will be spent on a software engineering project. However, this poses further complexities as there are no physical dimensions for software.

   The size of software to be developed is the main input parameter for estimating time/costs and thus must be estimated accurately. The relationship between the size of software and the effort required to produce is called productivity.

   Lines of code (LOC) is the principal PSP measure of size, but any measure which shows correlation to time can be used. The LOC size of code is the sum of added, modified or reused code, minus any deleted code. This can give us the physical size of the code, but is ill-disposed to estimating an engineer's productivity. Rather than measuring the size of the code to estimate time, we should be looking at the complexity of the task.

3. *Quality*

   Our finial and most important category of measurable data is quality- here we focus on anything that could detract from the program's ability to meet user needs.

   Defect Density refers to the defects per new and changed LOC in a program. We would like our software to never fail, but that is statistically improbable. It is important to look at the defect density, rather than the number of defects, when assessing quality as larger code leads to more defects.

   This metric runs parallel with the application crash rate- how many times an application fails divided by the number of times it is used. If your application crashes one time in a hundred but quickly recovers and doesn't lose any critical information, this defect may be acceptable. It is not just the number of crashes, but how quickly your application recovers from crashes that will affect quality. If the mean time to recover is much higher and crashes are costly the business to rectify, fixing it will impact the bottom line significantly.

   As engineer's design or code somewhere between 150 to 200 lines of code per hour, they may miss many defects. The Review Rate is the rate at which engineers can be expected to gather data and find all or most of the defects. An effective engineer will spend hours reviewing their code, searching for bugs and ways to improve quality.

## Drawbacks

Time can be a complex but critical element of data when exploring productivity. Should emphasis be placed on the time spent on each process, as an indicator of the amount of effort applied? Or would it be more beneficial to consider time as an adversary, where a more productive engineer is one which spends minimal time producing satisfactory products.

If engineers understand that productivity is related to time, could they not lie and say a project will take months longer to ensure a positive measure of productivity by meeting time constraints? Or conversely if engineers believe their goal is to have projects completed in minimal time, will this ensue a sacrifice of quality?

It is unreasonable to measure productivity based on LOC, as it discourages the reuse of existing, established and satisfactory code as a measure of increasing LOC. It cannot be

compared between engineers as it differs between languages. If an engineer believed their aim was to reduce LOC, they could implement measures the reduce their LOC, thus omitting necessary error handling.
It would seem the engineering process should be measured by the quality of the code produced. However, there must be a trade off between perfect code and the costs of testing and reviewing code. I would describe an efficient engineer as someone who takes pride in their work and ensures it is of the highest possible quality.

### Softer Data

I have recently completed a study on Margaret Hamilton, a remarkable software engineer who working with NASA on Apollo 11. It fascinated me to find out that software engineering wasn't her initial dream, she only began work with NASA to allow her husband completed college. Without her passion and focus on the project, I am certain that Apollo 11 wouldn't have been the tremendous success we remember it as today. It is my opinion that to successfully measure a software engineer's performance, job satisfaction must be considered.
Job satisfaction can be measured through interviews and questionnaires, and satisfied employees will have three traits:

1) Autonomy- the ability to direct his/her own work
2) Mastery- a strive for accomplishment/self-improvement
3) Purpose- knowledge that he/she is working on something more important than his/herself.

These are traits all displayed by Margaret during her work on Apollo 11. She ran her own vigorous testing and working alongside her team to produce piles of coding. She was constantly searching for new ways to improve her code, and often whilst drinking after work with her colleagues she would leave the bar to amend some piece of code. Hamilton's first concern was always the safety of the astronauts, and knew her code would extend into major future developments for NASA.

I am not suggesting that job satisfaction is the only data that should be used to measure a software engineer's ability, but rather it should be used in context with other measurable data to get the full picture of their work.

## Where to Compute

Once we have collected this data, it must be analyzed. Rapid analysis leads companies to more effective decision making, which can mean the difference between attracting customers or losing them to the competition. Developing analytics processes is not only work-intensive, but costly. It requires the purchase of different kinds of hardware, and the knowledge and expertise of IT staff who will implement and maintain these programs.

Analytics as a service (Aaas) refers to the provision of analytics software and operations through web-delivered technologies, and if used by business, AaaS could be used to bypass developing internal hardware for the purposes of business analytics. Providers of Aaas allow business to push data and offer access to an analytics dataset through a user-friendly interface for a fee. This allows the business to benefit from their services for a period of time, and cease payments later.

Not only are their significant cost implications, there is also a risk of compliance. Due to large wait times for enterprise data sets, business analysts were prompted to ask for more data than what they needed, and store this data on their personal computer for faster access. Analytics as a service can reduce compliance risks by allowing information workers to directly access data sets through an online portal. Mathiprakasam explains in his article "The Road to Analytics as a Service" that Multiple users can securely access the same information without having to duplicate data. Rather that replicating data for every user, they believe they have a personalized copy of the data. When users are finished with the datasets, they simply throw away their virtual access and no further cleanup is required. As all the data is stored on the cloud, the data remains physically secured and controlled by IT.

## Service Level Analyzation

As the risks associated with application failure have broad business impacts, the integrity of the APIs you produce and consume is more important than ever. Considering there are a great deal of companies that specialize in providing data analytics, one must be prudent in which corporation they select. You must keep in mind what exactly you are looking for, as it is futile to purchase a service we don't need or won't use. Areas of specialization include maintainability, code coverage, testing analysis and teamwork analysis.

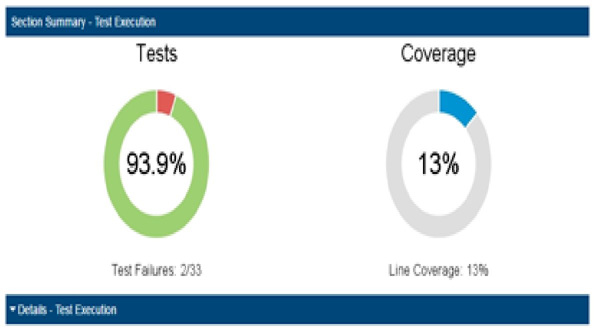
### *Unit Testing*

Parasoft is an integrated Development Testing solution proven to improve development team productivity and software quality. They provide end-to-end functional testing for complex applications, and have been a unit testing industry leader since 1997. Parasoft automatically analyzes code and generates a high-coverage test suite that is extensible and reusable. The main aim of this test suite is to expose functional problems and crash causing defects. Realistic functional tests are created via "tracing", a technique in which you mimic the application behavior. Automated distributions of tests are automatically sent to the appropriate code authors.

McCabe IQ follow the same approach to testing code. They believe that if you are not actively analyzing code through scrutinizing test activities using a path orientated approach, you cannot guarantee that your code will work the first time and every time. McCabe IQ have developed the Test Team edition, which assesses the thoroughness of your testing and establishes the resources to establish a well-tested application.

### *Code Coverage*

To effectively protect against bugs, a good software engineer will ensure they are testing as much of their code as possible. Every line of code has the power to improve, or reduce, the quality of your code. Code Climate, a software analytics company, are used by over 100,000 projects and analyze over 2 billion lines of code daily. Code Climate has incorporated fully configurable test coverage to ensure protection from bugs. This includes coverage of new code, per-file coverage and overall coverage tests.



Parasoft also specialize in code coverage analysis, ensuring test traceability requirements are achieved. Their service provides tests which measure code coverage to identify untested code and untested functionality. They also achieve test traceability to understand the impact of changes to code, focusing particularly on testing activities based on risk to meet compliance objectives. Along with a clear report of each test's pass or fail status, Parasoft report on the coverage for individual test cases, files, classes and functions.

A multi-metric test coverage analyzer reports coverage metrics as well as graphically displayed tested vs untested code.

### *Quality*

The quality of your code can be assessed through it's maintainability- how adaptable is it to new changes in software engineering? Code Climate scrutinize your data searching for duplication, analyzing complexity and examining the structure of your code to assess quality. Engineers can customize their test coverage target and to manage technical debt, meaning that you can fit the application around your specific goals.

CAST Application Intelligence Platform (AIP) is a software measurement and quality analysis solution designed to analyze applications for potential vulnerabilities and adherence to coding standards. This is done by assessing three key requirements of good measurement and quality including:

- Accuracy: CAST aim to uncover the true health of your applications by examining the quality of individual software components and their interactions.
- Precision: Identifying critical violations that deviate from good architectural coding practices, which could lead to stability, performance and data integrity issues.
- Standard: All analytics computed by CAST are built on industry standards. Parasoft have also implemented enforceable policies based on regulatory standards to ensure good quality code. This ensures measurements are comparable and comprehensible.

Business who are looking to assess a software engineers code would not invest in just one of these services, but perhaps a hybrid to ensure their specific needs are being met. The companies I have mentioned differ in various aspects, but a clear leader, in my eyes, would be Parasoft. They provide extensive data on almost all aspects of measuring software and have been industry leaders since 1997. The market has a lot of confidence in this company, with the majority of Fortune 500 companies relying on their services to produce high-end software consistency and efficiency.  Noted customers include Comcast, Lufthansa, Apple, AIG, Samsung and Coca-Cola, but to name a few.

## Risks of Analytics as a Service

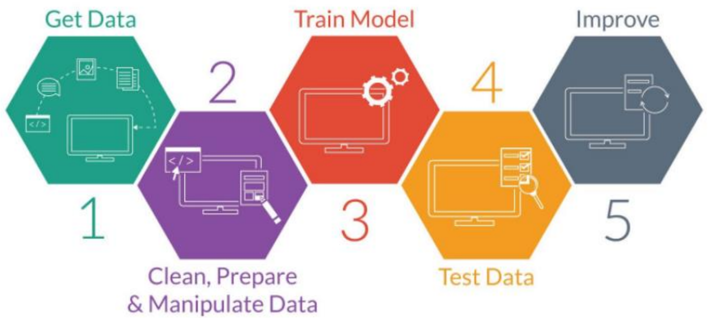
Clearly there are many benefits to employing Analytics as a Service to measure software, but every new concept comes with a certain level of risk.  There are concerns with the safety and privacy of data that is stored remotely. Undoubtedly there is a trade-off between communication/collaboration and security. You must ensure the service you use is secured and reliable. Developers and researchers must choose between easily obtained analytics and richer analytics with privacy and overhead concerns.

Although the platforms presented to analyze and measure the software engineering process are more than adequate, the way in which the software is analyzed seems to be extremely one dimensional. They allow easy instillation and integration to reduce costs to customers, but the results can provide little illumination of software processes and products. Even though the code we are analyzing seems to be scoring well on a technical basis, does this ensure it is achieving what we had set out to do? The approach doesn't support behavioral, client-side data collection and analysis and do not provide understanding into the use of developer practices (for example test-driven development).

## What Algorithms?

Over the last 250 years economic growth can be attributed to technological innovations. General-purpose technologies drive innovation and new opportunities, and undoubtedly the most influential technology of our ears is machine learning. This is the ability of a machine to think intelligently- to learn, apply reasoning and use self-direction Machine learning is rapidly replacing older algorithms, with an improved accuracy level standing at roughly 95%. Expert systems have been created where by computational learning devices can absorb new modification and put highly intelligent information together. Undoubtedly, new machinery has the capabilities to identify patterns in data more efficiently and quickly than humans can, which could prove extremely beneficial for our purposes of measuring software engineering as a process.

The majority of success in machine learning recently has sprouted from supervised learning systems, which provide machinery with many examples of correct answers to a particular problem. This generally involves mapping a set of inputs, for example data regarding an engineer's productivity, and generating predictions based on those inputs. These predictions can then be used to make personal recommendations to the user, in our case the engineer. Another machine learning algorithm is unsupervised learning, in which unlabeled data sets are sorted according to similarities or differences by the machine. There are growing opportunities in unsupervised learning, as machines will examine the data and attempt to establish patterns in engineer's behavior, allowing continued analysis into changes in productivity levels.

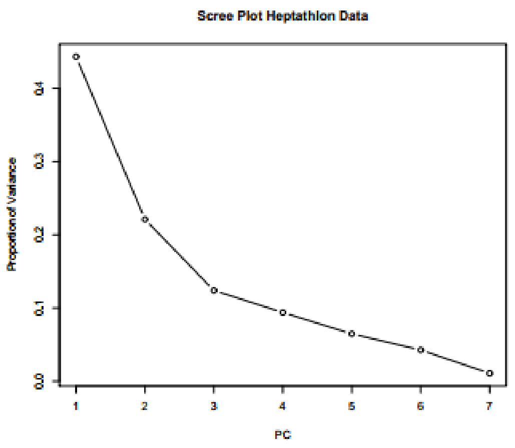


### Principal Components Analysis

As illustrated in the above diagram, before we can input the data into trained model it must be cleaned, prepared and manipulated. For data with many variables it can be difficult to understand inherent associations. Using PCA, an unsupervised learning technique, we can re-express the data with the aim of explaining its variation through linear combinations of the original data. A set of correlated variables is described in terms of a new set of uncorrelated variables, called principal components.

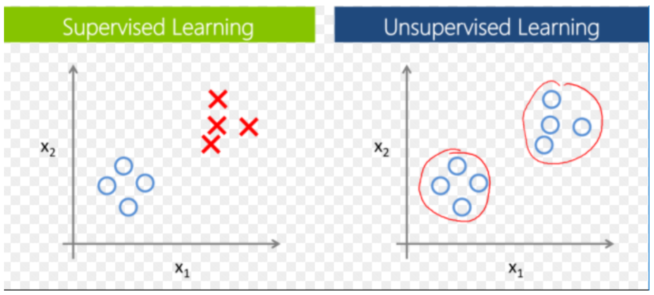
The aim of PCA is that most of variability will be explained by the first few principal components, allowing us to grasp which data surrounding an engineer's performance has the most effect on his/her productivity. Lower PC's provide less explanation about the data, and can be considered inherent noise. If we assume all of the data we collected has a strong impact on his/her performance, and use all of our data as input into machine learning algorithms, we could be modelling inherent noise.

There is no correct number of principal components to select to explain our data, but there are certain rules of thumb that can make this easier. If we set a target to have 90% of our information explained, then we should keep adding PC's until 90% of the variance is included. Another method would be to find a kind in a scree plot (variance explained against principal component number). Through a scree plot we can see the added benefit of including an additional PC may not be worth the cost implications of model complexity. As you can see on the scree plot provided the most efficient number of principal components to include would be 3, as after 3 principal components the plot levels off, and little additional variance is explained.



## Clustering Analysis

The aim of cluster analysis is to establish if there is a group structure in a dataset, and if so, how many exist and what are their individual structures. We would like observations within a group to be extremely similar, and to have different groups that are very dissimilar. Observations are placed in groups according to their similarity. First data must be standardized to ensure different variables are comparably scaled, this is done by dividing through their standard deviation before creating a dissimilarity matrix.



There are two types of cluster analysis methods commonly used, hierarchical and iterative. Both are types of unsupervised analysis whereby clustering is built up over several steps in which similar observations are joined. The groups are not predefined, and must be established during analysis.
Inputting the data selected from Principal Components Analysis into a clustering algorithm, such as k-means, the computer will predict and make comments on the performance of an engineer. By separating clusters into good performance and bad performance, this analysis should split engineers into both categories depending on their personal data calculated.

## Risks and Limits

Unlike humans, machines are not the good story tellers and it can be difficult to interpreting a systems output and understand how systems reached their decisions. Networks have many connections, each contributing a small amount to the ultimate decision. Although the machinery can give you the statistics on how an engineer is performing, it cannot tell you exactly why he or she was accepted for the job, or overlooked for promotion.
Although these tools are essential in understanding what makes a good engineer, by programming an algorithm to search for certain qualities in code/commits we are riddling the algorithm with hidden bias. Engineers may tailor their style of coding to suite exactly what the

algorithm is looking for, which could either help or destroy a company.

Another issue is that when the system makes errors it can be difficult to identify and correct what has gone wrong. However, machine based learning is still being improved over time. In my opinion, the next step in analyzing software engineering would be to develop theory of mind in AI algorithms. This refers to the understanding of beliefs, intentions and desires and can impact the way someone codes and makes certain decisions.

There is no denying that our future lies in AI, and for businesses to excel they must invest in data analytics and machine learning algorithms to understand what separates a good employee from a bad employee. However, human understanding and emotional intelligence must also be applied to a situation- perhaps a project failed due to dissimilar interest between co-workers? Perhaps an engineer underperformed due to reasons out of his/her control, for example illness? For this reason, we cannot solely rely on the algorithms discussed above.

## Ethics

The ability to collect and analyze data is moving much faster than any legal and ethical guidelines can manage. In most cases, businesses are collecting data for two purposes- to increase business profits and market more effectively and provide a more satisfactory customer experience. However, the methods in which data is collected may overwhelm users if they feel they are being watched and scrutinized. It is extremely important to abide by regulations surrounding data privacy, and ensure customers are aware of how any data they generate will be used.

### Unobtrusive Data Collection

Hackystat, a collaborative software Development Laboratory, utilizes unobtrusive data collection. This feature was developed upon discovery that developers found it is extremely frustrating to interrupt their work for manual data collection. To combat this, Hackystat ensured users didn’t notice that data was being collected, and any data collected while the developer was offline would be sent to the Hackystat repository upon reconnection. However, some developers consider this method of data collection as a bug and would prefer not to install instrumentation that would collect data on their recent activity and not report back to them.

The same thinking can be applied across all levels of data collection in businesses. In “The Trust Imperative: A Framework for Ethical Data Use”, Etlinger and Groopman explain that the first principle for ethical data use is that it should be done “with the expectation of tangible benefit”. Value should be delivered to both the individuals from whom the data is collected and the analysts. However, as soon as information is used to benefit the corporation and not the customers, or in this case the software engineers, they will bolt.

To illustrate this, I have two examples of how data collected from customers without their knowledge impacted on their experience with the company. The Ritz-Carlton are recognized worldwide for their top-class customer service and attention to detail. A classic example of this is when a waiter overheard a customer complain to her husband that she couldn’t go down to the beach, as there was no wheelchair accessibility. Upon hearing this the waiter immediately informed management, who arranged a ramp to be constructed by the following morning. Although the customer was delighted to have access to the beach, this information was overheard during a private conversation with her husband.

Contrasting to this, when Target attempted to track customer purchases and preferences to enhance the customer's experience the retailer's image was tarnished. Their custom advertising technology led them to send coupons to a teenage girl for baby products. Her father complained, only to find out that his daughter was in fact pregnant- news he only discovered because of Target's technologies. Their analysis was used to push sales and benefit their company, rather than deliver benefits to customers.

## Fine-Grained Data Collection

Another of Hackystats features is fine-grained data collection. Data is collected on a minute-by-minute or even second-by-second basis through a measurement called buffer transition. Data is collected when the developer changes the active buffer from one file to another. Hackystat can also track developers as they edit their code and construct tests. For this reason, users have coined the software ICU "hacky-stalk", complaining about the transparency it provided regarding the style in which each member works. For big data to work in ethical firms, information about how code is tracked and what exactly this is used for must be made more accessible.

It is my opinion that when analyzing a software engineer's performance using either of the two techniques mentioned above, it is important to consider how people would react if this information were out in the open- would it strengthen or threaten relationships? In my opinion, companies have the right to monitor their employees work to assess their performance. If an engineer is working on a project for a certain company, on hardware that is property of the company, why should this not be considered the company's data?

Undoubtedly, if results come back and insist a certain engineer is incompetent and incapable of carrying out the necessary tasks, further research should be done to discover the true reason they have not been prosperous. If it turns out that this is not a suitable occupation for a specific engineer based on the analysis, then this not only benefits the company by removing redundant staff but also benefits the engineer as they may move onto something they are more passionate about, and find more suitable employment. According to Frederick Taylor's "The Principles of Scientific Management", workers should be matched to their jobs based on capability and motivation, which would allow them to work at maximum efficiency. Monitoring performance should not be considered a negative, but a learning curve to enhance labor.

Of course, certain boundaries should be put in place to avoid invasion of privacy and misuse of data, and this has led to the development of laws and regulations regarding Data Collection.

## EU Data Protection Directive

It is vital that companies and websites that analyze data created by users make their privacy policy available. Customers must be alerted of any changes to this policy, how they can change their personal information and how the operator will respond to "do not track" requests.

This Directive applies to businesses based in the EU that are collecting the data of EU citizens. Its requirements include that:

- Data must be kept only as long as necessary
- Data must be collected for a specific, lawful purpose
- Data must not be transferred to a country outside the EU, unless that country provides adequate protection of the data.

As data collection capabilities grown, so do regulations. Soon the EU Data Protection Directive will be replaced by the EU General Data Protection Regulation, which will include stricter requirements on data collection notifications, new roles such as Data Protection Officer, and broader scope that applies to anyone collecting data, not just businesses.

It is up to CEO's to establish internal business ethics. It is vital that if management are collecting data on software engineering employees, they must have employee consent, or they will sacrifice employee trust and relationships.

### Overstepping Boundaries

In my opinion, ethics can be compromised by the data organizations choose to collect, and how this data can be manipulated to answer questions that could cause discrimination. If management are considering several applicants based on a dataset, perhaps one similar to what I have mentioned in section one, they have the ability to analyze this data and make inferences concerning race, gender and age. This use of data is extremely disrespectful and unfair.

It is not only illegal, but unethical and creepy to collect data online without the user's consent. In saying this, if corporations think they can gain valuable insights from tracking activity online, I have every belief they will do so. Long, complex privacy policies will deter users from investigating what exactly the company has access to, and is a form of manipulation so that corporations can access whatever data they want. In saying this, it is up to each and every individual to monitor their online activities- if they are behaving in antisocial conduct online there is always the possibility that this can be brought to light, as nothing on the internet is entirely private.

## Conclusion

The main objective of this report is to illustrate the need for both soft and hard data when measuring the software engineering process. When measuring hard data, to focus on the figures as a proportion of the overall size of the project. Figures can be manipulated and misleading, therefore non-quantitative knowledge such as employee satisfaction must also be employed in support of the data. By using principal components analysis, the most important variables are highlighted, which reduces costs as we only collect the data we require.

Companies that provide analytics as a service are the missing link that transforms figures and calculations to conclusions that can be used to better our business. One of the major benefits of using one, or many, of these companies is the ability to store your information on an online platform and access it whenever, wherever you are through a user-friendly interface. Such companies thoroughly asses code in areas such as unit testing, code coverage and quality, and can be "borrowed" for a certain period until their services become redundant, saving businesses time and money.

Different algorithms can give you contrasting results and conclusions, therefore the algorithm you pick will have a massive impact on how you perceive your data. As mentioned, data reduction is an extremely important first step in drawing conclusions. With too much information, we could draw conclusions that are based around errors. Artificial intelligence can reduce errors and lead to faster decision-making through automation, but this is at the cost of human intellect and reasoning.

In an ideal world, we could easily gather whatever information we wanted and apply algorithms to it without harming users or analysts. Realistically this is not the case, and data analytics has advanced at such a velocity that conclusions can be drawn on a person's sex or race based on how they code their data. This could lead to discrimination and other problems in the workplace, and feeds into how ethics plays a part in data analysis today. Every aspect of your life can be recorded through social media and your working habits, when does data collection over step personal privacy boundaries and become invasive? In my opinion, as long as company's follow the EU Regulations and ensure users know their work is being monitored, they are free to do what they wish with the information they receive. In any instance, knowledge is power and can only lead to improvements for both employee's and businesses.

## Bibliography

Akjnoukh, N. (2015, August 25). *You Can (and should) Measure Software Engineering Performance*. Retrieved from Kapost Engineering: http://engineering.kapost.com/2015/08/you-can-and-should-measure-software-engineering-performance/

CAST. (2017). *CAST Application Intelligence Platform*. Retrieved from http://www.castsoftware.com/products/application-intelligence-platform

Chafik, A. (2016, September 2). *How Much Data is Created Each Day?* Retrieved from Quora: https://www.quora.com/How-much-data-is-created-each-day

Code Climate. (n.d.). Retrieved from https://docs.codeclimate.com/

DevCreek. (2008). Retrieved from DevCreek: https://www.roseindia.net/eclipse/plugins/testing/DevCreek.shtml

Hamilton, L. (2016, March). *The Ethics of Data Analytics*. Retrieved from SessionCam: https://blog.sessioncam.com/the-ethics-of-data-analytics-579b942a78c2

Humphrey, W. S. (2000, November). *The Personal Software Process*. Pittsburgh, PA: Carnegie Mellon Softwaer Engineering Institute. Retrieved from The Personal Software Process.

Johnson, P. M. (n.d.). *Searching under the Streetlight for Useful Software Analytics*. Manoa: University of Hawaii.

Kassner, M. (2017, January 2). *5 Ethics Principles Big Data Analysts Must Follow*. Retrieved from TechRepublic: https://www.techrepublic.com/article/5-ethics-principles-big-data-analysts-must-follow/

Lowe, S. A. (n.d.). *9 metrics that can make a difference to today's software development teams*. Retrieved from TechBeacon: https://techbeacon.com/9-metrics-can-make-difference-todays-software-development-teams

Martin, E. R. (2014, March 27). *The Ethics Of Big Data*. Retrieved from Forbes: https://www.forbes.com/sites/emc/2014/03/27/the-ethics-of-big-data/#c8717fa6852e

Mathiprakasam, M. (2014, September 16). *The Road to Analytics As A Service*. Retrieved from Forbes: https://www.forbes.com/sites/oracle/2014/09/26/the-road-to-analytics-as-a-service/#7d44a9093622

McCabe IQ. (2017). *McCabe Application Lifestyle Management Solutions*. Retrieved from McCabe Software: http://www.mccabe.com/products.htm

Pal, K. (2017). *Advantages and Risks of Self-Service Analytics*. Retrieved from KDnuggets: https://www.kdnuggets.com/2016/04/advantages-risks-self-service-analytics.html

Parasoft. (2017). Retrieved from Parasoft: https://www.parasoft.com/

Richards, J. H. (2014, March). *What's Up With Big Data Ethics*. Retrieved from Forbes: https://www.forbes.com/sites/oreillymedia/2014/03/28/whats-up-with-big-data-ethics/#45c5a2953591

Roie, S. (2017, July 25). *Analytics as a Service*. Retrieved from Birst: https://www.birst.com/business-insights/analytics-as-a-service/

Rouse, M. (2016, December). *AI (Artifical Intelligence)*. Retrieved from SearchCIO: http://searchcio.techtarget.com/definition/AI

Slideshare. (n.d.). *Methods of Measuring Job Satisfaction*. Retrieved from SlideShare: https://www.slideshare.net/ShaolinRahman/method-ofmeasuringjobsatisfaction

Techopedia. (n.d.). *Analytics as a Service*. Retrieved from techopedia: https://www.techopedia.com/definition/29893/analytics-as-a-service-aaas

The Harvard Business Review. (n.d.). *The Business of Artificial Intelligence*. Retrieved from The Harvard Business Review: https://hbr.org/cover-story/2017/07/the-business-of-artificial-intelligence

TutorialsPoint. (n.d.). Retrieved from TutorialsPoint: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_overview.htm

Venturi, D. (2017, May). *Every Single Machine Learning Course on the Internet, ranked by Your Reviews*. Retrieved from freeCodeCamp: https://medium.freecodecamp.org/every-single-machine-learning-course-on-the-internet-ranked-by-your-reviews-3c4a7b8026c0

Weise, C. (2015, August 21). *4 Ways to Succeed with Analytics-as-a-Service*. Retrieved from data-informed: http://data-informed.com/4-ways-to-succeed-with-analytics-as-a-service/

Houlding B. (2017). *Logistic Regression*. Retreived from Scss.tcd.ie. [online] https://www.scss.tcd.ie/Brett.Houlding/ST3011_files/ST3011slides12.pdf

Houlding B. (2017). *PCA*. Retreived from Scss.tcd.ie. [online] https://www.scss.tcd.ie/Brett.Houlding/ST3011_files/ST3011slides3.pdf