

---

# Measuring Engineering – A Report

---

## 1. Introduction

---

The following report will consider the ways in which software engineering can be measured and assessed. I will address this topic through across several different headings including measurable data, the computational platforms available to perform this work, algorithmic approaches and the ethics surrounding this type of analytics.

## 2. Software Engineering Process

---

*“There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity” (Brooks F, 1987)*

To begin with, we must ask ourselves, what is the definition of software engineering? To put it simply, Software Engineering is an engineering discipline that is concerned with all aspects of software production. Software Engineering can be mainly concerned with the practicalities of developing and delivering useful software through fundamental processes such as software specification, software development, software validation and software revolution. But, where did Software Engineering come from? Software Engineering was first proposed at a conference organised by NATO in 1968 to try and deal with the ‘software crisis’ at the time. The software crisis was the name given to the difficulties in being able to develop large, complex systems at the time.

Since this time, the process of software engineering has certainly improved leaps and bounds in terms of the ability to develop large and complex systems. However, with different periods of time comes different challenges. As with the ‘software crisis’ briefly described there, we now face other problems in software engineering. Although it has a massive influence on the world we live in today, we must also learn to deal with the challenges posed. Fred Brooks understands that we cannot always see software development grow exponentially every year in which we ourselves might need a few years to solve. Over the course of this report, I will hopefully not just describe the challenges posed by software engineering but also the successes in the measurement and assessment of the software engineering process as a whole.

## 3. Measurable Data

---

*“Measures represent important data in all engineering disciplines”*

Why do we measure things? On a daily basis, people measure their shoe size, how tall they are and the weather. We consistently measure things in order to find how we’ve progressed in every step of life and this process is no different.

One of the most used resources in software is humans. Humans measure the data, manage the developers and write the sufficient code to enable the software. However, two parameter costs that go into humans are time and money. As the old saying goes, “time is money”, and

that saying is certainly relevant to the measurement and assessment of the software engineering process itself.

In Software Engineering, measuring useful data and collecting data is one of the biggest challenges faced by developers and managers. The difficulty behind collective measures is because

- 1) Collecting metrics is time expensive.
- 2) Manual data collection is unreliable.

Developers themselves do not consider collecting useful measures important in comparison to coding. However, blocking the inability for the evolution of software engineering, is the shortage of automated tools for collecting and analysing measures.

## Personal Software Process

Until the 1990's, most software engineering metrics were designed for use at organisational level rather than at individual. In 1995, Watt Humphrey published his book *"A Discipline for Software Engineering"*.

*"It would be nice to have a tool to automatically gather the PSP data" (Humphrey)*

Here, Humphrey proposed the *"Personal Software Process"*. His aim was to improve project estimation and quality assurance.

Humphrey's work was highly innovative in three main ways

- 1) Adapting organizational software process analytics to individual developers.
- 2) Showing how these analytics drove improvements.
- 3) Presenting the practises in such a way that it could be adopted by both academics and professionals.

Humphrey's can now be visualised along a timeline of generations with generation one being manual PSP, generation two being a toolkit called "Leap" and generation three implemented as "Hackystat".

| Characteristic      | Generation 1<br>(manual PSP)   | Generation 2<br>(Leap,<br>PSP Studio,<br>PSP Dashboard) | Generation 3<br>(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,<br>Context-switching | Context-switching                                       | Privacy,<br>Sensor availability |

Generation of approaches to metrics for developers


The Personal Software Process (PSP) provides engineers with a discipline personal framework for developing software. It involved manual input from the developers in forms such as

- A project plan summary.
- A process improvement proposal.
- A time estimation template.

These forms are only to name but a few. At each stage of their development, software engineers were able to analyse the inefficiencies that allowed them to improve their productivity through the planning, measuring and management of their own work. Their work was influence by the PSP data that supported both project estimation and quality assurance. To improve their own personal processes in the long run, they must meet each small term goal set allowed by identifying tracing their own performances, referring back to the planned schedule and identify elements that affect them. In the article “Searching under the Streetlight”, this part of the process was compared to lighting the candle. In terms of software engineering, this can be compared to the start of the process, but in order to consistently achieve quality products and reach high expectations, engineers must meticulously plan, measure and track product quality from the very beginning.

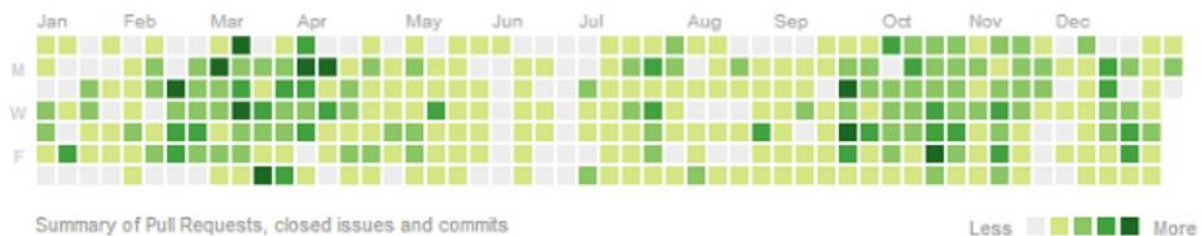
The second generation involved the implementation of “*The Leap Toolkit*”. The Leap toolkit attempted to address the data quality problems encountered with the PSP. By measuring, comparing and constantly improving, this generation allowed the automation and normalisation of the data needed to be analysed. Although the data is still manually entered, the benefit of the toolkit is to automatically generate analysis based on the inputted data. Leap allowed for the creation of repositories of personal transferable data, allowing developers keep their work from project to project and organisation to organisation. Software repositories provide a wide variety of information helping a developer to understand when and how a past improvement to their code has been changed. Leap also allows the developer’s names to be kept private by not referencing them in the data files. Metaphorically speaking, The PSP candle is now a campfire, introducing more support for the developer by improving data quality and decreasing the manual input and analysis from the developer themselves.

In addition, introducing “*Hackystat*”, a data collection tool developed by the University of Hawaii. After the data is collected, Hackystat allows for the automated analysis of the data second by second and minute by minute. The tool also provides for unobtrusive data collection, a feature important for developers as they attempt to increase productivity without interruptions and overheads. Both being key characteristics of Hackystat. Overall, the Hackystat built upon the original concepts of the PSP but with more advanced data such as code coverage and compelxity.

| 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  |  6.0  |  5.0   |  40.0  |
| duedates-omaomao (5)   |  64.0 |  1.2 |  6.2 |  1566.0 |  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 |

Software ICU display based on Hackystat.

Of course measuring your data is something that allows developers to compare, reflect and improve. If a developer's aim is to increase their own productivity, then measuring their own code is a necessary feature in doing this. For example, GitHub allows a developer to commit to their repository, create branches if they're working as part of a team, all while measuring the data uploaded and their performance in terms of software development. These metrics are essential in being able to improve efficiency but are not the fulcrum in keeping a software team together whilst working on a project.



Example of GitHub contributions

## Soft Measurable Data

---

On the other hand, while we can measure all the data in the world, what metrics do we have for measuring what an engineer's attitude, punctuating and overall health are in productivity. Although a developer may have many commits to a repository and is extremely efficient, it doesn't count for anything if they can't take criticism from their manager and don't show up to meetings or respond to emails. While their individual work is of a high quality, their contribution to the team they're working has a negative effect.

By looking at measurable data and progress in terms of systems and automation, it is clear that these techniques can certainly be used in being able to improve productivity while also seeing clear progression in terms of data.

In an article called "Measuring Happiness using Wearable Technology" reports that a person's happiness significantly influences performance. It has been found that people who are happy have 37% higher work productivity and 300% higher creativity.

Furthermore, the paper 'Network Effects on Worker Productivity' by Matthew J. Lindquist explores how co-worker productivity affects worker productivity via network effects. Data is collected over a two year period from an in house call centre and the results are a clear example of the opposite form of measuring data.

The results of study presented display that a 10% increase in average co-worker productivity produces a 1.7% increase in a worker's productivity. Furthermore, the addition of one extra co-worker to a worker's network increases that worker's productivity by 0.7%

But which way of measuring data is that correct method of implementing things? Is it necessary to constantly interrogate one's work for constant improvement by rigorously examining every line of code they write? Personally, I think too much of an emphasis is placed on this sort of data and only leads to an added pressure on a worker's shoulders who is faced

with many other challenges and pressures on a daily basis. More of an emphasis is certainly being placed on measuring worker's productivity as a whole and not putting an emphasis on the code they write. More measures are being put in place to allow worker's be happy in their workplace and network. By investing in this ethos, it is clear that a company will reap the rewards of having a happy workforce because the studies and results are there to show that they will receive in return an increased worker productivity as a result.

## 4. Computational Platforms

---

Knowing where to collect your data and the measures behind your data are key metrics in software engineering. It plays an important role in increasing the efficiency and productivity that allow a company to keep progressing in the right direction while also increasing its growth. We all know that how important data is in the world today, as we live in a revolutionary period in terms data analysis. It stands as the basis for most people's work if it's working as a data analyst, looking at the data behind your research project or simply reading the data as new reporter, no matter your work, you will always be surrounded by data.

*"Without big data analytics, companies are blind and deaf, wandering out onto the web like deer on a freeway" – Geoffrey Moore*

All companies, big or small, multinational or national, corporate or not, all have different ways to compute their data. The measures internally and externally between companies allow them to practise different measures, take different metrics and deliver their reports differently which makes why the presentation of everyone's data so unique. But, with all this talk about data analytics and the metrics behind it, what computational platforms do companies use in order to transpose this information?

## Code Climate

---

Code Climate is a data analytics company that specialises in the smaller pool of software engineering. Stated on their website, Code Climate uses the following description:



*"Code Climate incorporates fully-configurable test coverage and maintainability data throughout the development workflow, making quality improvement explicit, continuous, and ubiquitous."*

Code Climate is used by over 100,000 projects, and analysing over 2 billion lines of code daily. They speak the language of developers working the programming languages such as Java, Haskell, Python, Ruby and many more. By using code climates' services, companies can take control of their code quality through the automated code review for test coverage, maintainability so you can save time and merge with confidence. In software engineering today, time is one of the most scare resources available. Saving time will improve productivity for a company's developers and improve the overall software engineering process.

Technical debt is the term used to describe the extra development work that arises when code in the short run is easy to implement instead of using the best global solution. Code Climate allows you to combat the fight against technical debt. Frequently identifying changed files with no added value but that have inadequate coverage and maintainability issues, a company can reach its measurable goals. The software engineering process is a constant measurement and assessment of where you currently are and how you can improve to produce better results and reach higher goals and this aspect of code climate is an added incentive to make use of the services that they provide.

Code climate also collaboratively works GitHub to allow seamless integration between the two platforms. This collaboration allows for developers to get test coverage right every time, remove the pain from code review and identify hot spots like areas of high churn, for example. Developers can now work with full code coverage to avoid insufficiently merging, raising awareness about code quality while starting conversations on what matters to improve overall productivity. It is these measurement practises put in place by companies like Code Climate that are making the way in the next era of data analytics.

## Codacy

---

Codacy also presents an alternative to Code Climate setting the standard for automated code reviews. They provide the service of notifying developers on security issues, code coverage, code duplication, and code complexity in every commit and request. Developers workflow is seamless integrated between codacy and platforms like GitHub and Bitbucket. Codacy like Code Climate allows you to improve productivity by controlling your technical debt and focusing on your work that matters more. Codacy provides its services to major companies like Paypal, Adobe and Deliveroo.

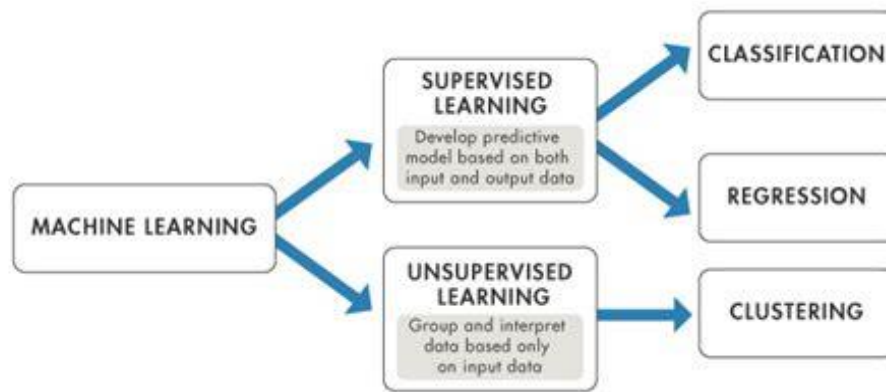


## 5. Algorithmic Approaches

---

In terms of measuring and assessing the software engineering process, our report up to this point has looked in depth into measurable data and platforms to compute this. When we collect data, what do we do with the data? What algorithms are there to analyse this data? How do we use these algorithms to analyse the data? The field of data analytics is one that is becoming increasingly popular but how can we leverage the data obtained to gain the most insightful answers from the questions we have on the presented data.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning is often categorized into supervised and unsupervised learning, the details of which I will explain now.



## Supervised Learning Machine Methods

---

Supervised learning follows the guidelines where the algorithm receives a set of inputs along with the correct corresponding outputs and the algorithm learns by computing its comparing its actual outputs with the correct outputs to find errors. This can be used to measure the productivity of an engineer who may map a set of inputs where the algorithm then makes predictions based on those inputs in order to try and improve the algorithm for future use, therefore bettering the engineer. Examples of supervised learning techniques include classification.

Linear Discriminant Analysis (LDA) is an example of a classification technique. LDA refers to a set of supervised statistical techniques where the class information is used to help reveal the structure of the data we are dealing with. It therefore, allows the classification of future observations of the labelled data on additional unlabelled data. This fundamental statistical method can be visible in many of today's machine learning methods.

## Unsupervised Learning Machine Methods

---

Unsupervised learning is used where unlabelled data are set according to similarities or differences by the machine. The goal of the machine is to explore the data and find some structure within while also drawing inferences from datasets to describe hidden features from unlabelled data. An engineer's work can be examined to try and find patterns and make recommendations upon this in order to increase efficiency. Examples of unsupervised machine learning include Principle Components Analysis and K-means clustering.

Principal Components Analysis (PCA) is a dimension reduction technique. PCA is extremely useful through its ability to compress a dataset with many dimensions into something that captures the essence of the original data. It can be thought of as a method for re-expressing the data so as to reveal its internal structure and explain its variation through the use of a few linear combinations of the original variables. A set of correlated variables is described in terms of a new set of uncorrelated variables, called principal components.

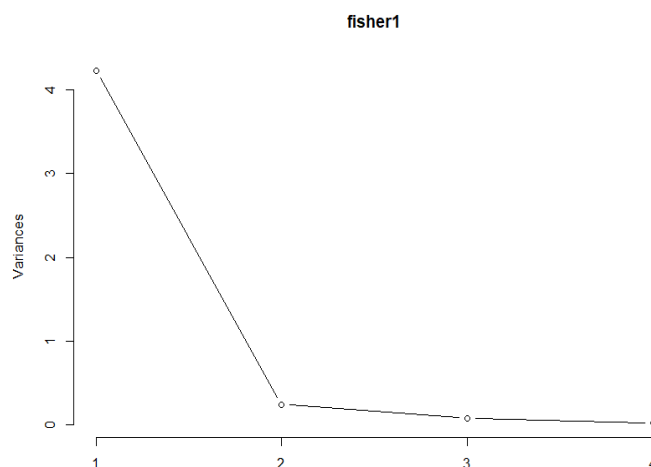
The aim of PCA is that most of the variability will be explained by the first few principal components. The proportion of variance explained by each PC aids the choice of the number of relevant components to include. Preferably, three principal components would be the most PC's used to explain a set of data. While performing PCA, there is a trade-off between the



complexity of the analysis and ensuring you capture enough variance of the dataset presented. In reference to an engineer's performance, his/her performance would be explained more in the first few PC's as it will capture most of the variability of the data due to principal components being ordered in terms of importance showing a reduction in variance as we increase the number of dimensions.

There is no correct number of principal components to use to explain the sufficient level of variance to cover in any data set. However, there are a few rules of thumb.

- 1) Keep adding PC's until a fixed proportion of variance is included.
- 2) Find a kink in a Scree plot. This will show the correct number of principal components to use. In the below plot, we pick two principal components which covers 97.769%. After PC2 on our Scree plot, we can see that the added benefit of



including an additional principal components is not worth the extra cost of model complexity.

## Limitations of Machines

---

Recently, I competed in a competition called “Accenture Solves”, where the aim of the competition was to try and come up with a solution to allow the upskilling of the workforce who will lose their jobs to Artificial Intelligence in the years to come. However, machines lack the art of storytelling as it can't be classified as a science. Storytelling can't be replicated, it is unique in its beautiful, magnificent way. When I think of AI, I think of optimism and pessimism. Pessimistically, yes AI is almost certainly going to cause many people to lose their jobs. For example, look at Amazon Go, the first fully automated shopping market where people who stack shelves and cash registers are no longer needed. However, on the other hand, because of the AI, new work will arise that only humans do. It is a case of glass half empty or half full.

## 6. Ethics

---

Having already explored the topics of measurable data, computational platforms and algorithmic approaches in measuring and assessing the software engineering process, I will now examine the ethics regarding these practises. Ethics is the discipline dealing with what is good and bad and with moral duty and obligation. How controversial or not, some issues that will be raised, it is necessary to dissect and investigate them for both their good and bad effects and features. It is important to note, while inspecting this topic, we must remember the words in the definition of ethics, “*moral duty*” and “*obligation*”. Do the companies looking after the data of their workers and customer have a moral duty? And do they obey to the laws and regulations in place? Because after all, it is their obligation to.



Every day, everywhere we go and everything we do is being monitored and collected as data. As a generation, our use of social media platforms is now greater than ever especially Facebook. Facebook keeps a record of our data but also reaches out to other social media platforms when you login into their accounts using your Facebook account. However, earlier this year it was released that data consultancy company, Cambridge Analytica, gained access to millions of Facebook users accounts and then used this information for political advertisements, in no other than the 2016 US Presidential Campaign as well as the Brexit referendum campaign to name but a few. The number of Facebook users affected is estimated around 87 million people.

After reports of the data breach hit the headlines, Mark Zuckerberg, CEO of Facebook had to face the US Congress to explain his company's case that they had in fact told Cambridge Analytica to delete the data but they had failed to meet their obligations.

As a result, the EU began to change its own regulations regarding data protection. As of 25<sup>th</sup> May 2018, the EU replaced its Data Protection Act (DPA) with the General Data Protection Regulation (GDPR). These new laws and regulations ensure stronger rules on data protection allowing people to have more control over their personal data and businesses will also benefit from a level playing field. It is vital that all companies including those in the software industry oblige to these new regulations. Failure to comply with these new regulations will result in fines up to €20 million or 4% of company's worldwide turnover. The new regulations are making waves across all sectors, not just multi-national companies but small and medium enterprises have an obligation while also trying to combat the risks that GDPR hold. After all, the law is the law and they are put in place for a reason.

## Personal Opinion

---

Personally, I feel that measures like GDPR are put in place to protect us as citizens as a whole. As I mentioned above, the definition of ethics explain how an obligation is involved. We as people have an obligation to protect ourselves and our own data, companies have an obligation to protect their employees and their own data and companies have an obligation serve their customers and users and protect their data.

Earlier this year, Mark Zuckerberg said in front of the US Congress,

*"We have a responsibility to protect your data and if we can't, we don't deserve to serve you"*

When the new regulations were first announced regards to GDPR, they seemed to be a phase of fear and craze in response. It made me think to myself, did people not care about the DPA beforehand? Were companies obliging to original regulations and rules? What does GDPR mean now for everyone? In my opinion, privacy has a major role to play in terms of protection. We would never give information to a stranger without consent, therefore, why would we give away all the data about ourselves while using online platforms that are a major part of our lives.

However, when consenting in the use of your data, we are placing a sense of trust in major companies that we are confident that they will hold their moral duty and obligation to you higher than they hold ability to abuse the data. This is a difficult concept to consider, especially when you think that sometimes that you can't trust your closest friends or loved ones. So how can one trust companies like Facebook to keep their confidential information? If they have given your data away before, who is to say that they won't do it again? Personally, I think if someone doesn't want to give their consent, then they shouldn't, and they shouldn't give their consent without considering the implications.

With regards to software engineering, it is essential that ethics plays a role in the measurement and assessment of the process. A developer's employer or manager should not collect the data of its employees without using it in a beneficial manner that will improve performance and the overall process and values in which a company operates in. It is important to motivate a worker from collecting the correct data and using it in a beneficial manner that will only instil confidence in an engineer's ability.

It takes someone to stand up to unethical issues in terms of data collection and be a whistleblower. Someone like Edward Snowden, who was brave, courageous, and held his moral obligation above his own standard of life. We need more people who have the ability to stand up and say no to people who have no obligations to serve the people.

## Conclusion

---

In conclusion, it is clear to see that the software engineering process can be analysed, dissected and interpreted from many different viewpoints, stances and opinions in terms of measurable data, computational platforms, algorithmic approaches and ethics. However, one thing that is common among the headings in which we discussed the process of software engineering process is that each one faces their own challenges and in overcoming those challenges have improved the measurement and assessment in how engineers work.

In every system, whether it be in terms of measurable data, computational platforms, algorithmic approaches or the ethics behind collecting data, engineers will always be at the epicentre of the process. It is their duty to ensure that the measurement and assessment of the software engineering process learns from its failures. As the saying goes, you must fail to succeed.

## Bibliography

---

- [1] [https://www.youtube.com/watch?v=Dp5\\_1QPLps0](https://www.youtube.com/watch?v=Dp5_1QPLps0)
- [2] [https://www.researchgate.net/profile/Jan\\_Sauermann2/publication/285356496\\_Network\\_Effects\\_on\\_Worker\\_Productivity/links/565d91c508ae4988a7bc7397.pdf](https://www.researchgate.net/profile/Jan_Sauermann2/publication/285356496_Network_Effects_on_Worker_Productivity/links/565d91c508ae4988a7bc7397.pdf)
- [3] [http://www.hitachi.com/rev/pdf/2015/r2015\\_08\\_116.pdf](http://www.hitachi.com/rev/pdf/2015/r2015_08_116.pdf)
- [4] P. M. Johnson, H. Kou, J. Agustin, C. Chan, C. Moore, J. Miglani, S. Zhen, and W. E. J. Doane, "Beyond the personal software process: Metrics collection and analysis for the differently disciplined," in Proceedings of the 25th international Conference on Software Engineering. IEEE Computer Society, 2003.
- [5] A. Sillitti, A. Janes, G. Succi, and T. Vernazza, "Collecting, integrating and analyzing software metrics and personal software process data," in Proceedings of the 29th Euromicro Conference. IEEE, 2003, pp. 336– 342.
- [6] <http://csdl.ics.hawaii.edu/techreports/2012/12-11/12-11.pdf>
- [7] <https://codeclimate.com/about/>
- [8] <http://www.citeulike.org/group/3370/article/12458067>
- [9] <http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/no-silver-bullet.pdf>
- [10] [https://www.sas.com/en\\_us/insights/analytics/machine-learning.html#machine-learning-importance](https://www.sas.com/en_us/insights/analytics/machine-learning.html#machine-learning-importance)
- [11] [https://www.scss.tcd.ie/Brett.Houlding/ST3011\\_files/ST3011slides3.pdf](https://www.scss.tcd.ie/Brett.Houlding/ST3011_files/ST3011slides3.pdf)
- [12] [https://www.scss.tcd.ie/Brett.Houlding/ST3011\\_files/ST3011slides4.pdf](https://www.scss.tcd.ie/Brett.Houlding/ST3011_files/ST3011slides4.pdf)
- [13] <https://www.cnet.com/news/facebook-cambridge-analytica-data-mining-and-trump-what-you-need-to-know/>
- [14] [https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes\\_en.pdf](https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf)
- [15] <https://www.slideshare.net/software-engineering-book/ch1-introduction-42645973>
- [16] <https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/History/>
- [17]

Taghi Javdani, Hazura Zulzalil, Abd. Azim Abd. Ghani, Abubakar Md. Sultan, On the current measurement practices in agile software development, International Journal of Computer Science Issues, 2012, Vol. 9, Issue 4, No. 3, pp. 127-133.

[18] <https://www.codacy.com/>