Measuring Software Engineering

**Foreword**

The following report has been compiled to offer an insight into the measurement and assessment of the software engineering process, specifically, relating to its measurable data, computational platforms, algorithmic approaches and ethical concerns. This report has been influenced by a selection of scholarly papers, books and online material largely concerning the task of measuring workers in order to review and improve performance.

**Software Engineering Overview**

Software Engineering can be described as a process that addresses the theories, methods and tools for specialised software development [1] and with the ever increasing number of software controlled systems the measurement and assessment of the process has never been more important. Furthermore, just as in the classical physical engineering process, software engineering also embodies the design, construction, maintenance and testing of a process. Before investigating the measurement and assessment of the software engineering process it is helpful to have an idea of the benefits and applications of the process; It involves recognising complex problems and finding an optimum solution to them. To find a solution to a problem is often not the largest problem that a company may face, it is finding the optimal solution that can prove to be so troublesome. However, there is no one optimal solution in software engineering, this varies depending on what the company prioritises, be it time taken, code coverage, complexity etc. these are some of the topics that will be addressed in this report.

**Measurable Data**

Sillitti et al tell that a great software engineer’s productivity is “ten times better” than the average engineer’s [2]; However, this is a meaningless statement without being able to acquire meaningful, measurable data. A commonly held belief is that quality is intangible and while it is something that could be observed it could not be measured. In contrast to this the view of the professional software engineer is that quality can be both measured and analysed; however, the main debate does not question the ability to measure and analyse data but instead whether these analytics “can deliver meaningful results” [3 ]. These opposing beliefs stem from the fact that quality is not a single, universal concept but instead a “multidimensional” [4] idea.

The most basic measurement of product quality is to look for the presence of bugs; the defect rate and reliability of the software can be tested to see how many errors a system possesses and how often it exhibits failures. The table below exhibits the methods used by two leading software companies to monitor the quality and satisfaction of their products: [5]

|  |  |
| --- | --- |
| **BM** | **Hewlett-Packard** |
| **C**apability  **U**sability  **P**erformance  **R**eliability  **I**nstallability  **M**aintainability  **D**ocumentation  **S**ervice  **O**verall | **F**unctionality  **U**sability  **R**eliability  **P**erformance  **S**erviceability |

The similarities visible in the assessment methods above suggest that there is an agreed upon standard measure of quality in the software engineering environment; Not only do developments have to be capable but also lasting, readable and easily maintained. Real world examples have other more obvious features that must be considered in the software engineering process; the measurements mentioned previously have to be balanced against the costs and time required to obtain the desired result and a compromise chosen.

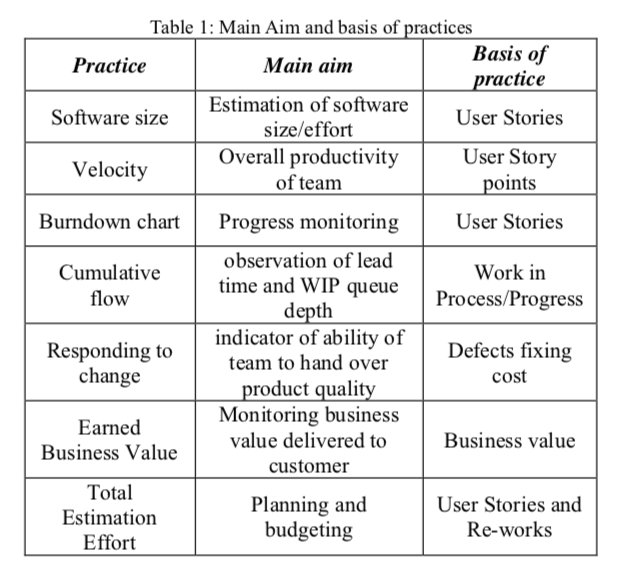
The software engineering process could be categorised by four levels of measurement: the nominal scale, the ordinal scale, the interval scale and the ratio scale

Nominal is a simple form of measurement such as classification which places components into categories based on a specific attribute. The nominal scale is often employed as it is a simple, easily understood method of date management, however, is a limited scale as it does not offer a ranking of the categories components.

Ordinal scale is a form of measurement in which the components can be compared and furthermore be ranked. Nevertheless, again it has limitations, namely that it can rank components but does not offer information regarding the difference between components.

Interval scale provides the exact difference between various quantifiable components making it a much more useful measurement process.

Ratio scale is one in which it is possible to locate an absolute or nonarbitrary zero value. It is the uppermost form of measurement processes and is suitable for any mathematical operations. For example, it would allow a software engineer to label one program’s fail rate as twice that of another’s.



The figure adjacent displays the aims and basis of practice for a series of types of measureable data, further illustrating the variable nature of measuring the engineering process.

[6]

It could be said that there are three conventional software metrics: size, cyclomatic complexity and comment percentage; the size of an algorithm can be a valuable metric as it offers an insight into the readability of the development and how easily interpreted it can be by other engineers. The cycolmatic complexity can be calculated by creating a control flow graph of the code used and finding the number of linearly-independent paths through a program; the lower the cyclomatic complexity, the easier the algorithm is to understand and the lower the risk is when making modifications. The final metric, comment percentage parallels the number of lines of code with the number of lines of comments; a good software engineer will add regular, explanatory comments to ensure an understandable and maintainable program.

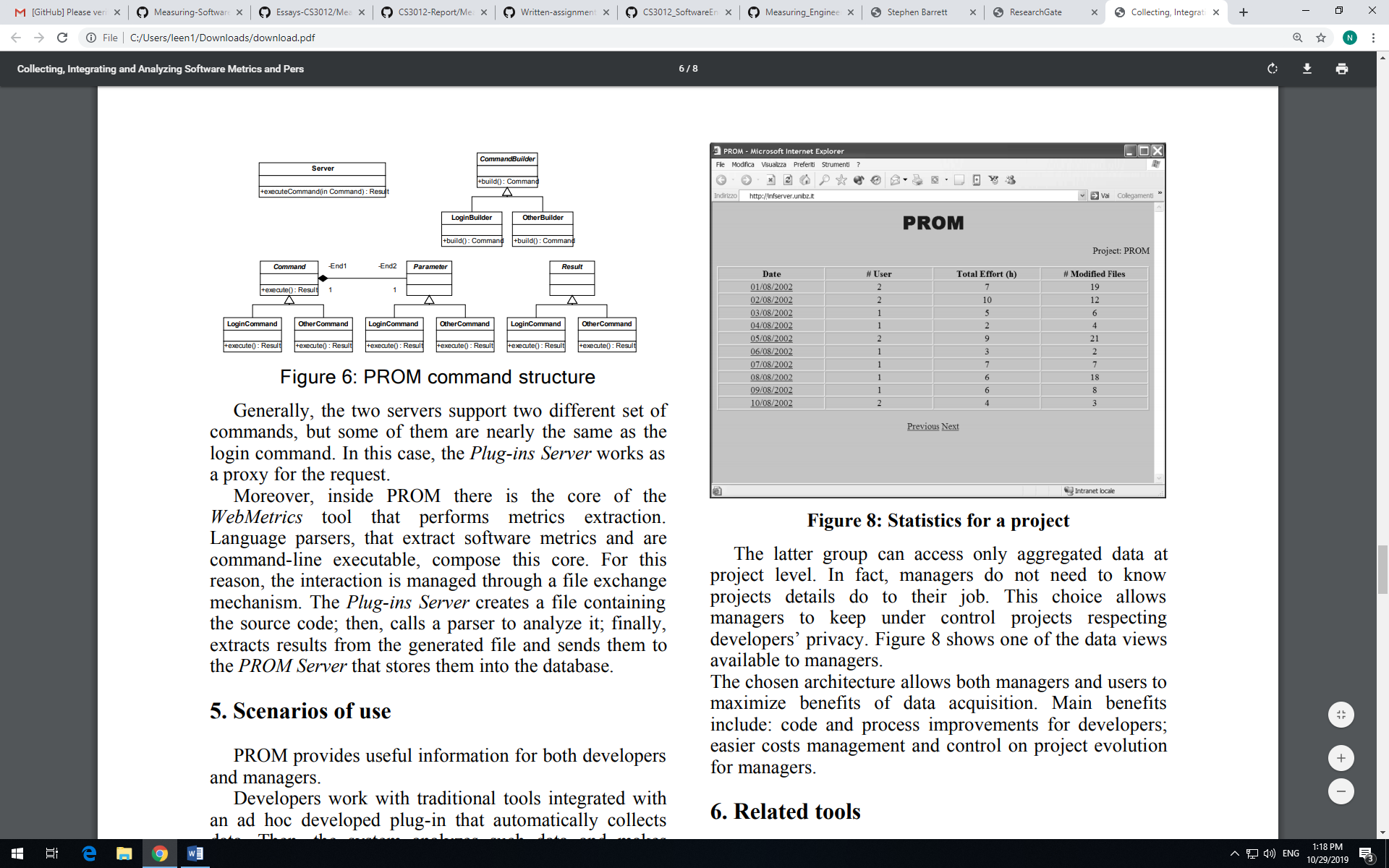
**Computational Platforms**

Personal Software Process (PSP)

Watts Humphrey aka the father of software quality [7] developed the Personal Software Process, an innovative metric that looked to measure and analyse the software engineering process, supporting project estimation and quality assurance. Humphrey argued that if an engineer could acquire a knowledge of the PSP that the results would be evident in their future developments; The PSP offered a platform that could aid the planning and tracking of an engineer’s work. Furthermore, the system helped establish goals and analyse the engineer’s performance accordingly. The PSP revolved around self-assessment; the user would record the size, time and defect data from their work and the process could then offer analysis on such entries such as the estimated project time. While a revolutionary process at the time of conception, the PSP’s scalability was limited due to the laborious need for data entry, which also left the process very susceptible to human error. Nevertheless the PSP was to pave the way for future computational platforms.

Pro Metrics (PROM)

While PSP had displayed the benefits of data measurement and analysis it was still viewed as time consuming and inconvenient; many software engineers believed their time would best be spent developing rather than focusing on self-assessment and analysis. PROM offered an automated solution to this issue that could gather software metrics without the need for data entry by the engineer. PROM is both useful for the individual engineer or a manager analysing the efficiency of a project group; this allows for code and process improvements for engineers alongside cost and efficiency management for managers. An example output of the PROM system is shown below: [8]



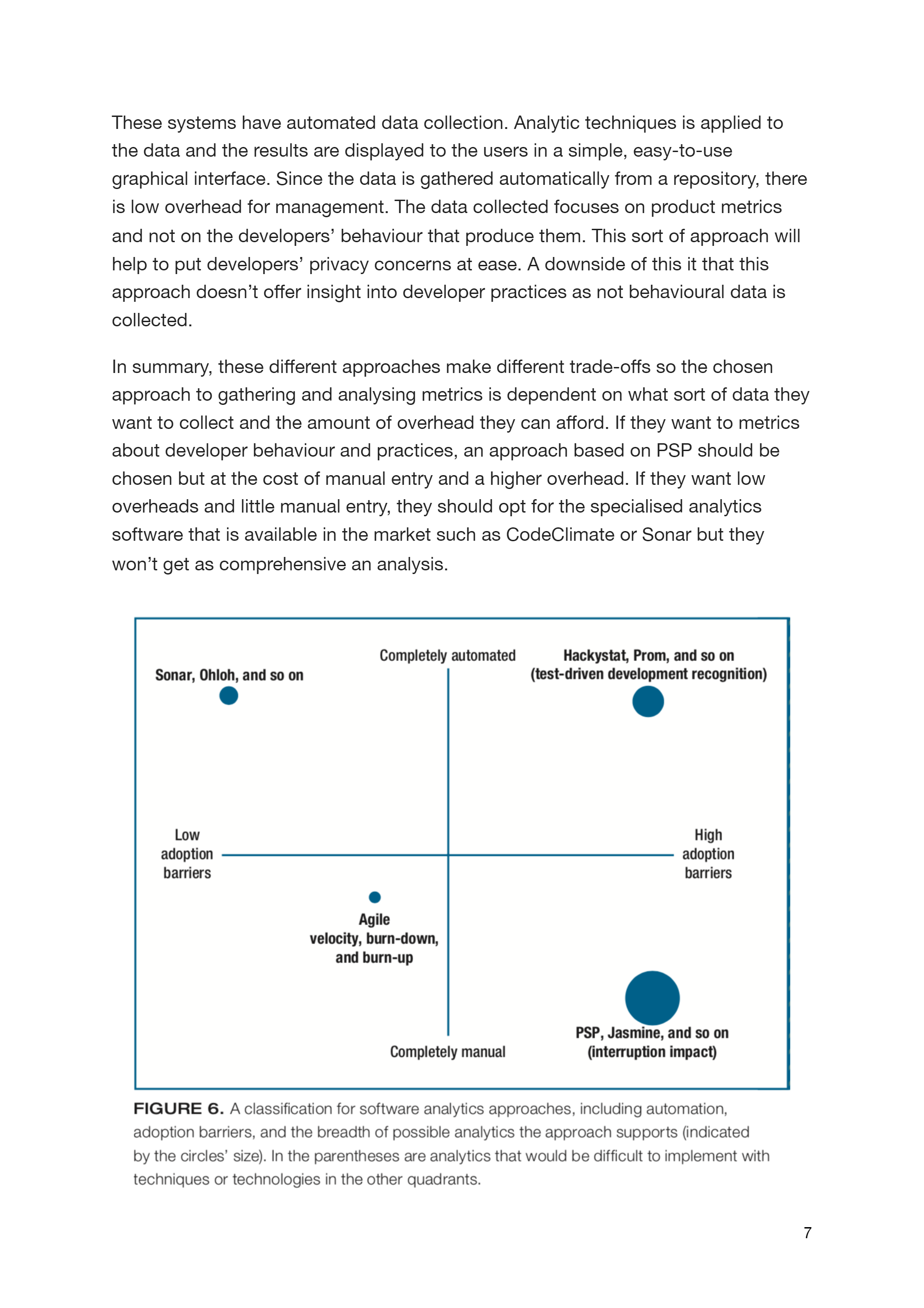
Leap Toolkit

Leap was developed as a comprehensive system for collecting and analysing software metrics, similar to the PSP but with less demanding overheads in data collection and analysis. It offered an automated analysis system, however, while less demanding it still featured aspects of data entry by the user; in Leap the user must regularly tell the tool what they are working on and the system will then process analysis on this work. Despite significantly reducing the overhead and human error associated with PSP, Leap was not the comprehensive, trouble free system that software engineers were looking for.

Hackystat

Hackystat was created in an endeavour to counter the problems that had limited PSP and Leap; the development finally offered a comprehensive, fully automated data collection and analysis platform. The solution was found by adding sensors to development tools; the data is then transported to the server where analysis is carried out and any resulting alerts are sent back to the user. Despite the elimination of the data collection and analysis overheads, Hackystat still received mixed reactions in the software engineering community; some users were displeased by the unobtrusive form of data assembly employed by Hackystat as their performance was being monitored without their knowledge. Similarly, some discontent was reported by users who were not comfortable with other group members and management having such detailed access to their workings.

The information above offers a brief overview of some of the foremost computational platforms used in the software engineering environment; perhaps the most significant observation is the greatly varying opinions on each of the platforms. Every engineer and every project will hold different aims and tendencies and the choice of computational platform is usually selected in a compromise between the detail of analysis desired and the available overhead. The figure below offers a visualisation of this compromise: [9]



**Algorithmic Approaches**

In a similar style to the progression of computational platforms over the years, the technological advancements in the software engineering environment have allowed for massive progressions in the algorithmic approaches to data measurement and analysis. The increasing efficiency of algorithms alongside the consolidating power of machine learning have vastly increased the scope of data analytics without the difficulties of large overheads and lengthy processes.

Computational Intelligence

Many of the algorithmic approaches to data measurement in software engineering are centred around computational intelligence, which describes a machine’s potential to learn and improve processes based on historic data and experimental observation. Computational intelligence consists of a host of smaller sub-categories which combine to observe, approximate, model and evaluate data:

Neural Networks (NNs) are used to process, select and send information; NNs allow for a machine to learn and generalise based on similar, existing algorithms.

Fuzzy Systems are used to overcome uncertain or incomplete problems by adopting a generalised version of traditional logic; these results can then be applied to approximate reasoning.

Evolutionary Computation generates, evaluates and modifies a series of potential solutions and uses these processes to overcome optimisation problems that were not solvable using a mathematical approach.

Learning Theory is an essential component that approximates reasoning; it is a useful tool in comprehending the processing of cognitive, emotional and environmental effects and then computing predictions accordingly.

Probabilistic Methods are used to gauge the results of systems based on randomness; they offer potential solutions to reasoning problems, constructed using historic experience and information.

Machine Learning

Machine learning has extraordinary potential, including the capability of grouping and interpreting data and producing predictive models; the process can be broken down into supervised learning and unsupervised learning.

Supervised Learning

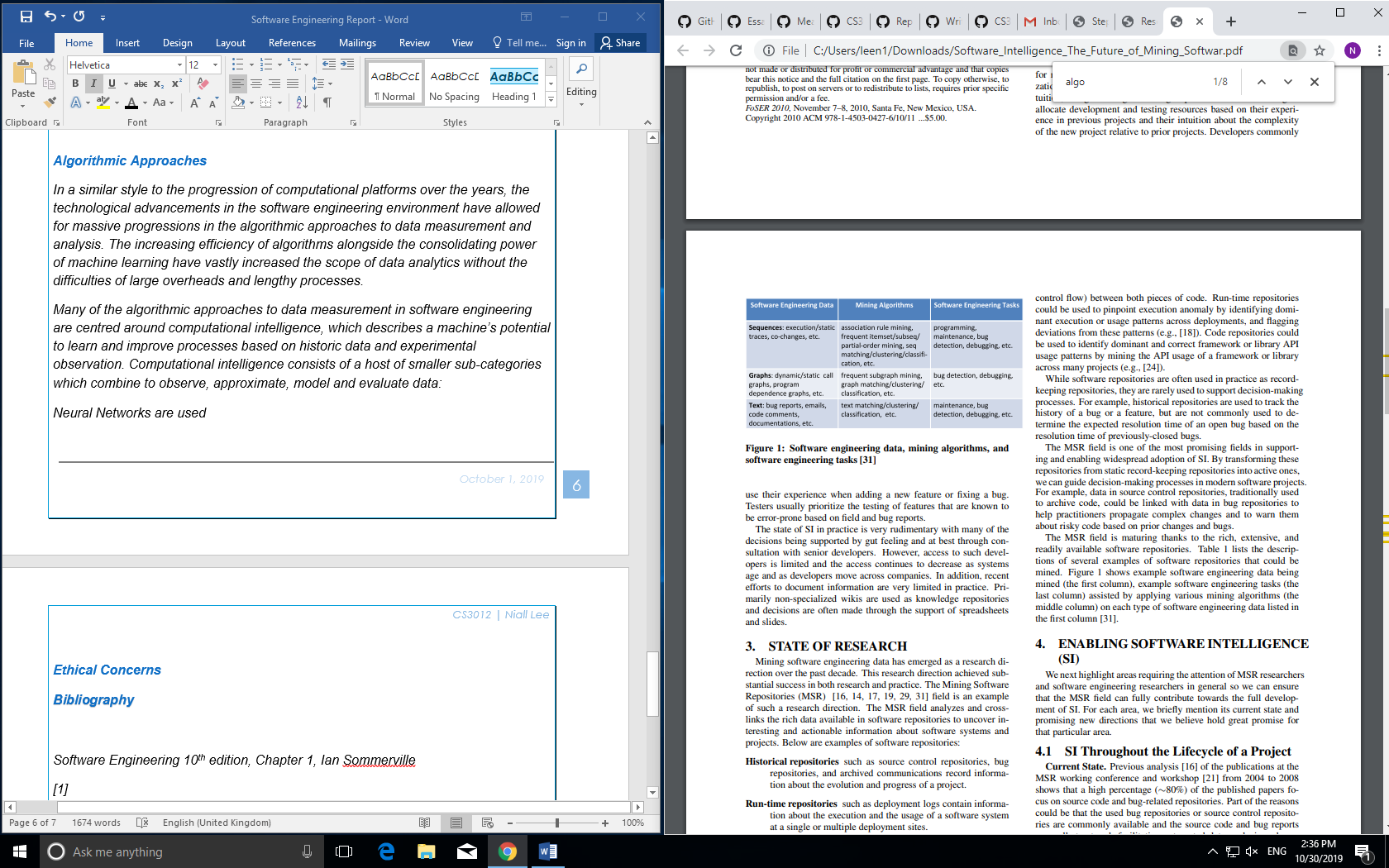
Supervised learning encapsulates most machine learning methods; it features methods containing x input variables and a y output variable that are acted upon by learning algorithms to learn the mapping function between inputs and outputs. In supervised learning methods any particular input will have a known output and it is therefore easier to identify prediction errors.

Examples include: Linear regression for regression problems, Random forest for classification/regression problems, Support vector machines for classification problems. [10]

Unsupervised Learning

Unsupervised learning methods differ from supervised methods in that an output data set is not obligatory. They are of use when an engineer faces an unlabelled data set, the algorithm can find inferred relationships between data points. The algorithms involved operate by grouping data in relation to their properties, similar and different.

Examples include: K-means for clustering problems, Apriori algorithm for association rule learning problems [11]



Further information on mining algorithms in regard to the software engineering environment [12]

Process Quality Index (PQI)

The process quality index is a statistical algorithm capable of measuring the accuracy of a process’s output production; it can provide estimates of how close an engineer is to any selected target and how much they deviate from their usual performance. The method defines five quality component values and finds their product to determine an overall quality index for any particular process, these quality component values are highlighted below: [13]

|  |  |
| --- | --- |
| Design Quality: |  |
| Design Review Quality |  |
| Code Review Quality |  |
| Code Quality |  |
| Program Quality |  |

To calculate the PQI the five ratios are multiplied together to produce a value between 0.0 and 1.0; the higher the PQI value the higher the quality of the process. Furthermore the value of each individual component can be compared to recommended industry values or historic data to help identify problems in the process.

Review

Despite the confronting beliefs within the software engineering community I believe that algorithmic approaches and in particular machine learning and computational intelligence undoubtedly merit a place in the industry. The increasing capabilities and efficiency of machine learning mean that their applications to measuring the software engineering process are only going to increase. However, I do agree with David Parnas who argues that artificial intelligence will not provide a revolutionary advancement in measuring software engineering quality [14]; this belief stems from the attempt to solve problems with machines using a human interpretation. Nevertheless, the ability of these processes to take in, manipulate and analyze massive quantities of data has in many ways exceeded human capabilities and if further progress can be made in replicating human empathy and creativity their potential is limitless.

**Ethical Concerns**

Data measurement and analysis is a day to day regularity in many businesses across an array of industries, ranging from the assessment of an individual’s performance to the monitoring of an entire corporation. In large, the intentions behind this measurement is innocent, usually with the ultimate goal of increasing company efficiency and worker productivity. However, when a company is gathering data of any form from its employees there are serious ethical considerations, particularly when the measurement is automated. There exists issues, legal and ethical in many aspects of the process including the type of data collected, the processes involving the data and the continuous storage of the data.

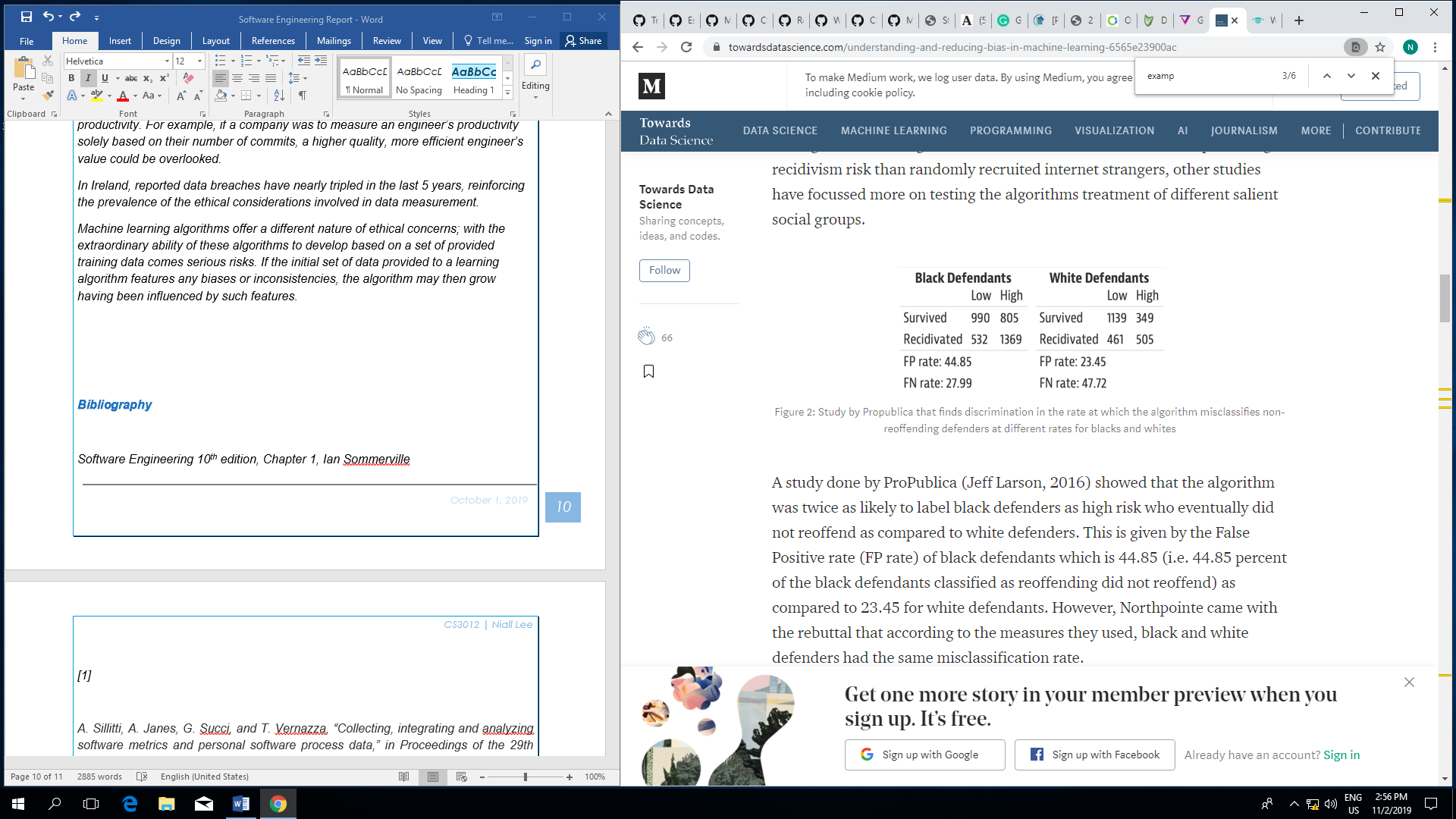
Examples of recent legislation:

* In the UK, under the Communication Data Bill, Internet Service Providers are required to keep a log of any site accesses and e-mails by users dating back one year. This has raised significant controversy on where to draw the line between safety (tracking criminal activity) and privacy. [15]
* In France, recent legislation sanctions government agencies to block a user’s internet connection if they have accessed illegal content using peer-to-peer networks. [16]
* The most glaringly obvious change was introduced by the European Union in May 2018 [17]; The General Data Protection Regulation (GDPR) was implemented to improve data protection standards for users and consequently crack down on companies who infringe on these sanctions. The GDPR addresses the measurement of both personal data (name, number, account details etc.) and of special category personal data (ethnicity, religion etc.); special category data cannot be processed without explicit prior consent from the user unless special authorization by law is granted. [18]

It is my belief that data measurement very much merits a place in the software engineering environment so long as its reasoning is sound and its implementations are fair. If a company chooses to collect and store the data of its employees or projects it must be directly related to the employee’s work and it is imperative that an employee’s personal privacy is not infringed. Furthermore, when gathering employee data I think that the entire process must be as transparent as possible, informing the employee exactly what data is being collected and why. “Why” is also a central consideration; while the collection and analysis of data can provide useful insights to employers such as employee working hours or sales etc. in other industries its uses are not as obvious. In relation to software engineering, tracking an employee’s quality or efficiency is a difficult process as discussed previously; with this in mind employers must be cautious of rewarding or punishing an engineer’s recent productivity. For example, if a company was to measure an engineer’s productivity solely based on their number of commits, a higher quality, more efficient engineer’s value could be overlooked.

In Ireland, reported data breaches have nearly tripled in the last 5 years, reinforcing the prevalence of the ethical considerations involved in data measurement. [19]

Machine learning algorithms offer a different nature of ethical concerns; with the extraordinary ability of these algorithms to develop based on a set of provided training data comes serious risks. If the initial set of data provided to a learning algorithm features any biases or inconsistencies, the algorithm may then grow having been influenced by such features.



This figure illustrates the racial discrimination present in the COMPAS algorithm, showing the differing rates of the algorithm misclassifying reoffenders alongside the defendants race. [20]

The American judicial system offers an example of the potential dangers of learning algorithms; an algorithm known as the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) algorithm is used to aid judges in selecting the duration and type of criminal sentences. The algorithm was engineered based on the results of a 137 piece survey and an enquiry by ProPublica’s Jeff Larson found that black defendants were twice as likely to be labelled ‘high risk of reoffending’ as white defendants. [21]

****Another high profile visualisation of data measurement’s ethical sensitivity came to light in 2018 in the Facebook-Cambridge Analytica data scandal. This scandal saw the user data of approximately 87 million Facebook users unknowingly shared with a British political consulting firm [22]; the breach saw the user’s data used for targeted advertisements and was used in part to further Donald Trump’s presidential election campaign. In the modern day digital world that we live in, user data is more valuable than ever and I therefore believe many aspects of data measurement such as this case to be exploitation; if money is to be made off a user’s data is it not reasonable for them to have ownership or alternatively expect compensation?

In many ways a reinvention of the software engineering environment as a whole is necessary as a consequence of the changing digital world. Future software engineers must widen their scope, a one dimensional though process of software creation is no longer viable; in my opinion, for a software development to be truly successful it must look further than bugs or failures and encompass sound ethics. The task of assigning responsibility and liability for data measurements is difficult and the future is sure to see changing attitudes and increasing regulation on the subject.

**Conclusion**

Measurable data is an asset that can be found in every aspect of software engineering and the quantities of data collected are only likely to vastly increase. However, with the difficulty of defining quality in the industry it is imperative that companies sufficiently research before implementing data collection and analysis; the process must be moulded uniquely to the company for it to provide value to the company and furthermore, misuse of the process can lead to a counter-intuitive effect and potential legal dilemmas. Ultimately, the outcome will emerge from the management of the measurement process; with careful selection of computational platforms and monitored use of learning algorithms there exists the potential for huge development in the industry. Of course the increased adoption of data measurement and analysis goes hand in hand with the ethical concerns, however a great engineer has the ability to come up with innovative solutions and it is this that reinforces my faith in the possibilities of data measurement.

**Bibliography**

- Software Engineering 10th edition, Chapter 1, Ian Sommerville

[1]

- 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.

[2] [8]

- 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.

[6]

- Stephen H. Kan. 2002. Metrics and Models in Software Quality Engineering (2nd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[4] [5]

- Hyatt, L. and Rosenberg, L. (1997). Software metrics program for risk assessment.

- ["National Medal of Technology Winner Watts Humphrey, 1927 - 2010"](http://www.sei.cmu.edu/newsitems/Humphrey_obituary.cfm)

[7]

- “What Makes a Great Software Engineer?” Luo LI, Ko & Zhu

- ”Searching under the Streetlight for Useful Software Analytics” Philip M. Johnson

[9]

- Hassan, A.E. and T. Xie, Software intelligence: the future of mining software engineering data, in Proceedings of the FSE/SDP workshop on Future of software engineering research2010, ACM: Santa Fe, New Mexico, USA. p. 161-166.

[12]

- PSP(sm): A self-improvement process for software engineers, Watts Humphrey

[13]

- “Software Engineering Ethics in a Digital World”, Rashid, Weckert & Lucas, IEEE

<https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/>

[10] [11]

<http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/no-silver-bullet.pdf>

[14]

<https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/index.html>

[3]

<https://www.citizensinformation.ie/en/government_in_ireland/data_protection/overview_of_general_data_protection_regulation.html>

[18]

<https://www.independent.ie/business/technology/news/data-protection-cases-soar-three-times-as-many-breaches-reported-to-commissioner-as-five-years-ago-37866508.html>

[19]

<https://www.theverge.com/2018/1/12/16882408/google-racist-gorillas-photo-recognition-algorithm-ai>

<https://towardsdatascience.com/understanding-and-reducing-bias-in-machine-learning-6565e23900ac>

[20] [21]

<https://www.siliconrepublic.com/companies/facebook-media-currency-digital-ethics>

<https://www.bloomberg.com/news/articles/2019-09-09/facebook-users-gain-leverage-in-cambridge-analytica-privacy-suit>

[22]

<https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/228824/8359.pdf>

[15]

<https://techcrunch.com/2007/11/23/if-you-p2p-download-in-france-no-internet-for-you/>

[16]

<https://gdpr-info.eu/>

[17]