**Measuring Software Engineers: A Report**

*By Aoife Kelly*

The software engineering process can be measured a number of ways. One may choose to focus on analysing the performance of an individual engineer or the output of a team. There are numerous different types of measurable data and subsequent computational platforms to analyse this information. However, the increased focus of data collection and algorithmic based decision making have led to many questioning the ethics concerning these analysis. This report aims to provide an overview of these topics with a conscious focus on the issue of how ethical these practices are.

**What Data?**

So how does one quantify what makes a good software engineer? It is not simply one factor that separates the elite from the masses but a combination of elements. The same can be said for a whole host of professions. For example, a footballer is not deemed the best simply because he scores the most goals, other factors such as speed, endurance, possession and team work need to be taken into consideration. But what factors make a software engineer effective and are they measurable?

One easily measured value is the number of lines of code written by the software engineer over a given period. However this has been found to not always be the most accurate method of assessing skill. One particular problem can have numerous different possible solutions and researchers have found there to be up to ten times the variation in code volume among different designers dealing with the same task. Using lines of code per staff over a given time as a measure of productivity would implicitly suggest that a programmer who writes ten times the amount of code as another programmer to solve a particular problem is more productive. This is a statement which goes against one of the main goals in software engineering: to write code as concisely as possible, all the while maintaining a semblance of readability and clarity of thinking. Great programmers don’t necessarily always write the least amount of code. They always write clear code which often results in less code. Often in order to write clear and simple code a little more lines are needed than a design that’s more "clever". Sometimes less is more, sometimes more is more but when it comes to programming the most is nearly always a waste of time. Which leads to the obvious conclusion that while number of lines of code per programmer can be measured easily, should one even bother as it is clearly not an accurate measure of skill. (Construx.com, 2017)

Should we therefore measure function points in order to ascertain the productiveness of a software engineer? Some of the issues associated with using lines of code per programmer outlined above can be avoided by measuring program size in function points rather than lines of code. Function points are a method of measuring program size. Inputs, outputs, queries, and files are counted to determine the size of the program. Inefficient coding won’t generate more function points, so function points aren’t subject to the same issues as lines of code. However function points do have some drawbacks. In order to obtain an accurate count of function points one needs the services of a certified function point counter, which a considerable amount of organisations don’t have. Also the mapping between how function points are counted and individual work packages is rough enough that it becomes impractical to use them to gauge the effectiveness of individual programmers. (Construx.com, 2017)

Complexity is another aspect it is important to consider. When given a complex piece of code to work on, a skilled programmer may take the same time to complete this task as a fellow employee who is tackling a much simpler problem. By some measures of productivity it may appear that both programmers have the same level of ability. This is of course untrue as the second worker may have taken twice as long to complete the more complex task and thus measurements should be taken to decipher the complexity of the code being written. Whilst this is a more difficult value to measure than lines of code written per person or function points, it is arguably a more accurate reflection of skill. (Construx.com, 2017)

Then there is the aspect of the volume and severity of bugs present in the code written. The number of bugs at any time can be measured and dealt with accordingly. Companies such as Kapost Engineering, have different priority levels for bugs, and set thresholds at each level. Consequently, they measure themselves by how well they are keeping to those set thresholds. The severity of the bugs is also an issue. It is important to look at how many users are reporting bugs. A single quite severe bug will consequently be reported by many users. It is a good idea to measure severity relative to the number of active users. For example, ten bug reports from one million active users is outstanding, conversely ten bug reports from ten active users is terrible. Measuring both volume and severity provides us with a clear overall picture of the state of bugs. (Akhnoukh, 2017)

As the popular phrase goes “it’s not quantity but quality”, so there for code quality must come into consideration when measuring the effectiveness of a software engineer. How much of the code is being tested? It’s all well and good to write a thousand lines of error free code but how many of those lines are being tested? Tests are an integral part of software development, all aspects of code must be tested in order to ascertain that code is working as intended. Therefor test coverage is another important measure of software engineering. A high test coverage implies well thought out, quality code written by a thorough and talented programmer. Whereas a low test coverage score gives the impression that the code was hastily thrown together without much thought and consideration.

We must also take into account more abstract measures when attempting to gauge a software engineer’s effectiveness. Business impact for example is another vital element of the software engineering process. Great running code is often frivolous without business impact. Programmers are employed to write code for a specific purpose, for example to increase the number of users or to drive better adoption of the system. While these goals should be the main focus of the product management team, a healthy engineering team is highly aligned with the product team, and consequently should also consider these goals when writing code. Programmers should constantly remember to be aware of what the bottom-line impact of their work has on the business. Google use measurements such as this as a way to access those eligible for promotion and worthy of bonuses. Coding is often pointless if it doesn’t serve to impact a business or society in any way.

The ability to work in teams is a quality sought after by numerous multinationals. In a survey with the National Association of Colleges and Employers (NACE) in 2015, companies Chevron, IBM and Seagate Technology were among the 260 participants who declared teamwork as the most valuable skill they look for in an applicant. (Poole, 2017) Many of the greatest success stories in the world of technology have been developed not by a singular person but a group of individuals working towards a common goal. Google was founded by both Larry Page and Sergey Brin who later recruited Eric Schmidt to help in the development. Max Levchin and Peter Thiel were the duo behind PayPal. (Successstory.com, 2017)These successful software engineers all recognised the value of collaboration and had the ability to work with others. What is the point in having numerous perfectly crafted pieces of code if the contributors are incapable of pulling them together to a working solution? However how do we measure a person’s capability of functioning in a team environment? Peer reviews seem to be the obvious choice but that means leaving the concrete realm of numbers and facts and delving into the land of opinions and potential bias. Information such as numbers of lines of code, test coverage and bugs are far easier to collect with less potential for errors.

The question of what data to use is one with countless answers, some more relevant than others. The answers above are just a few of the ways employers can and are monitoring the effectiveness of their employees. As explained each of these methods have their own individual drawbacks and there for should be used with caution. But regardless of what data we choose to use, we must decide what to do with all this information collected. There is also the issue of how various companies intend to harvest this data, with some methods more intrusive than others.

**Where to Compute?**

The problem of where to compute all this Big Data is a hot topic in our current market. Many companies are springing up with the main objective on capitalising on the growing market for data analytics. Progression in methods of data collection have led to a mass amount of data available but this data is useless if not studied and used for methods of improving efficiency and overall productivity.

The Personal Software Process (PSP) is one method of analytics used to observe software engineering. Whilst it can provide users with deep insights into the programming process, it often leads to significant overhead cost for developers. There are several different versions of PSP, although most rely heavily on manual input of data which is usually time consuming. For example, one version of the PSP, requires users to fill out 12 forms, including

• A project plan summary,

• A time-recording log,

• A defect-recording log (see Figure 1),

• A process improvement proposal,

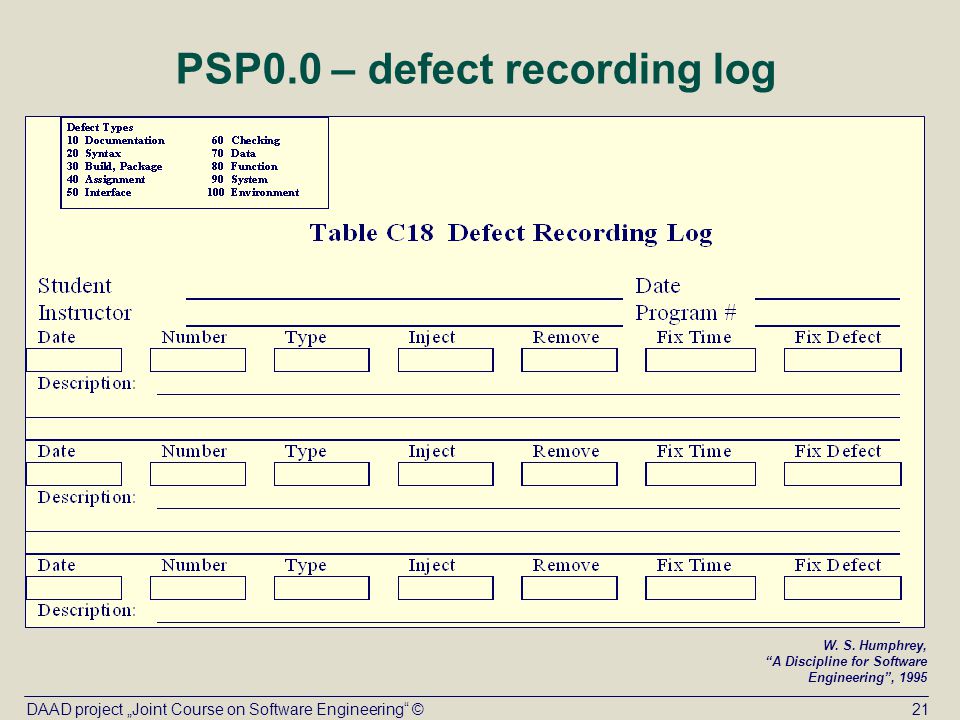
• A size estimation template,

• A time estimation template,

• A design checklist, and

• A code checklist.

These forms usually contain over 500 distinct values which developers must manually calculate. Using PSP developers can modify the forms and procedures to address specific circumstances and needs they feel are of interest. The manual nature of the PSP makes its analytics flexible but also fragile. It allows its users to search for the analytics they feel are important to improve their efficiency. However the potential for human error is significant which results in issues with data quality. In a study carried out by the University of Hawaii in which they checked over 30,000 data values generated by classroom use of the PSP, they found that its manual nature could result in incorrect process conclusions despite a low overall error rate (less than 5 percent). As it is not possible to fully automate PSP, which results in the need for significant manual data entry, many believe that it doesn’t provide enough return on investment. Especially considering historical data is not suitable for comparison among different projects. (Johnson, 2013)



*A sample PSP defect recording log*

In response to the difficulties faced by PSP, the University of Hawaii developed Hackystat. It “implements a service oriented architecture in which sensors attached to development tools gather process and product data and send it to a server, which other services can query to build higher-level analyses”. One interesting feature of Hackystat is its emphasis on unobtrusive data collection. One of the main complaints concerning manual data collection is the interruption it causes. Developers are forced to continually stop working on their code in order to record what they just worked on. With Hackystat users are unaware the data is being collected. Hackystat is able to track a programmer as they edit a method, construct tests and invoke said tests. It also works for groups of developers, for example tracking the interplay among them when they edit the same file. However Hackystat has run into some issues among its users, primarily with programmers being uncomfortable with the way data was being recorded automatically without their knowledge. They were also unhappy with management having access to the data collected. A main principle of agile software development is to “build early and often.” Hackystat has the ability to measure how well developers adhere to this and management have this information at their fingertips. What was developed as a method to save developers time and prevent interruptions soon became viewed as some sort snitch, breathing down the necks of those it was intended to aid. In the University of Hawaii’s pursuit to fix the issues raised by PSP they encountered a whole host of other problems. (Johnson, 2013)

Code Climate, who label themselves as “superpowers for engineering teams”, are yet another organisation who aim to take a bite out of the data analytics apple. It is used in conjunction with GitHub in order to assist programmers on a more individual level. It provides users with the information necessary to ensure they are merging code which is clean, healthy and sufficiently tested. Target test coverage is fully customisable for each programmer and users are able to decide which technical debt issues should be considered blockers. This level of flexibility is one of the reasons Code Climate is so popular among developers and used by over 100,000 projects worldwide. There is also an emphasis on personal development with progress reports provided which show the most important changes that were merged and a trends page detailing if overall code quality is getting better or worse. In addition, Code Climate helps software engineers see which projects require their attention by correlating code quality information against areas of high churn and thus identifying those with inadequate coverage or maintainability issues. (Codeclimate.com, 2017)

So how do Hackystat and Code Climate compare? From my understanding, Hackystat is comparable to that irritating child who would inform the teacher that you didn’t do your homework. Code Climate on the other hand is that friend who helped you work out that maths problem you just couldn’t get your head around. If you have your homework done constantly then you realistically have nothing to fear with Hackystat, but are you comfortable in the knowledge that someone if constantly checking that it is? Conversely, Code Climate works with the user providing aids that allow them to better their work and clearly see how their projects are coming along. It measures the developer against themselves, not its fellow staff members.

Analytics as a Service (AaaS), the latest buzz word born from this data analytics craze, is another method used to compute the data measuring software engineer’s performance. It combines “the on-demand aspects of cloud computing with the democratization of information enabled by big data analytics”. AaaS allows enterprises to innovate continuously and make more real-time decisions. It eliminates manual IT tasks that have prevented business users from gaining access to technology services. More importantly it does all this whilst ensuring the enforcement of security, privacy, and compliance controls. Analytics as a Service allows workers personalized access to centrally managed information data sets, removing painstaking delays that were previously present. As a result one can explore information data sets more interactively and thus discover richer insights at a more rapid pace. In the past, business analysts would ask for more data than they needed due to long wait times for enterprise data sets. This surplus data was then stored on their personal computers which became a growing cost problem and a compliance risk. AaaS was the answer to this growing problem. If analysts need data, they simply access a web portal to request a personalized data sandbox with the data needed from the master data warehouse. Once they have completed their analysis, they release access to their data sandbox back to IT. As a result of cloud computing, it is not necessary to physically make a copy for every user who wants to view the data. When finished with the data required users just relinquish their virtual access and no other clean-up is required by IT. No data traces remain on the worker’s computer since all of the data is only stored in the cloud. This allows IT to accelerate access, save costs, and maintain forensic control over data. With faster access to information for staff that AaaS offers, tech companies have the opportunity to evolve into real-time enterprises. All the data collected about software engineers can be easily analysed leading to a deeper understanding into the insights the information provides. (Mathiprakasam, 2017)

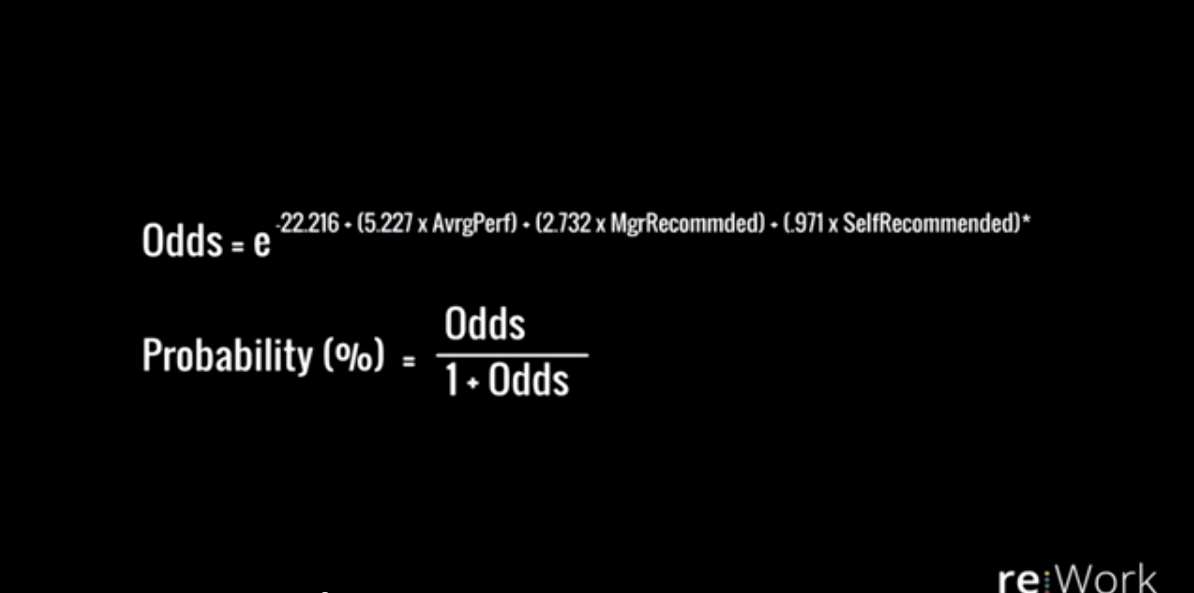
PHP, Hackystat, Code Climate and AaaS are just some of the many methods available for computing data related to the productivity of software engineers. When choosing which method to apply companies must consider how much flexibility they require with their analysis. Another point up for consideration is what level of scrutiny are software engineers comfortable with? Intrusive processes such as Hackystat are not viewed positively among certain programmers due to the high level of data being collected without their permission. All methods of computation have their own benefits and drawbacks, the search for the perfect solution is still ongoing.

**What Algorithms are Available?**

What algorithms can then be used to measure the performance of a software engineer? All this data collected can be used in a multitude of algorithms, which provide with a deeper understanding of the information. However we must approach these kinds of analysis with an air of caution and not be too reliant on their results in our decision making.

Google have invested significant amounts of time and money into the creation of algorithms to study their staff members. In 2007, they created a People Analytics team whose main role was to collect and use data to improve the company’s management practices. Google executive Prasad Setty claimed that “all people decisions at Google should be based on data and analytics”. Setty concluded that Google “wanted analytics to spit out people decisions.”

They chose to take a highly data-centric approach to the hiring of new employees. However Google soon found there were limits to the power of algorithms and the subsequent automation they desired. They encountered this issue early on in one of the first missions for the People Analytics team. The group formulated an equation to assist Google in deciding which of its software engineers were worthy of promotion. This was a decision previously made by committees of senior Google engineers. The algorithm the People Analytics team developed is shown below.



Studies show that the algorithm was capable of making a third of Google’s promotion decisions with 90% accuracy rate. It was also viewed as reliable and stable across multiply cycles. Thus through the power of algorithms and analytics the team were able to decrease hiring committees workload by a third, a resounding success one might think. However that is not how things panned out. Whilst the algorithm was accurate, the engineers simply hated it and it was never used to make any promotion decisions at Google. The senior engineers didn’t want to hide behind a model and let it make decisions for them, they wanted to make their own decisions and involve human elements in the process. After this experience the People Analytics team came to a realisation that decision making would never be a completely automated process and that people were needed in the making of people decisions. The groups focus then became to better equip decision makers with useful information, not to replace them with algorithms. (Nisen, 2017)

Computational intelligence can use algorithms to make inferences about the productivity of software engineers. It can be defined as the study of the design of intelligent agents. An agent is a fairly broad term and is effectively something that acts within an environment. An intelligent agent more specifically is a system which acts intelligently. It is capable of acting appropriately for its circumstances and its goal whilst also being flexible to changing environments and goals. Intelligent agents are further able to learn from experience and make choices given certain limitations and finite computation. The central scientific goal of computational intelligence is to understand the principles that make intelligent behaviour possible, in natural or artificial systems.

Artificial intelligence (AI) is the more commonly known name for the field just defined as computational intelligence. However the question repeatedly being asked is artificial intelligence real intelligence? To answer this question which must consider how we define intelligence. The goal of AI is not actually to simulate intelligence but to understand real intelligent systems by synthesizing them. The most obvious intelligent agent is the human being. However there are a class of intelligent agents that could potentially be more intelligent than humans, the class of organizations. Ant colonies are an example of this. An individual ant may not be described as intelligent, but an ant colony is capable of acting more intelligently than any individual ant. A colony is able to discover food and exploit it effectively all whilst adapting to changing circumstances. A team of software engineers can similarly develop software where the sum of the skills required is more than an average individual could understand. Computers have become more complicated than can be understood by any human but are manufactured by teams of humans. Human society as an agent is arguably the most intelligent agent known. (Cs.ubc.ca, 2017)

Algorithms have changed the way we make decisions based on the overwhelming amount of data it is possible to collect. They allow businesses to make more informed choices and aid them in making the correct decision. However increased reliance on algorithms should be met with a healthy level of trepidation. When assessing the productivity of software engineers, a human element which is absent in logical algorithms, is necessary. The main takeaway is that people decisions need to be made by people not computers.

**A Question of Ethics**

With all this information freely available at employers fingertips, one is forced to ask the question, is any of this ethical? Where is the line drawn between the goal of trying to achieve increased productivity and respecting the privacy those in employment. Does privacy even really truly exist anymore in our everyday world? Personal information has become increasingly available through our increased reliance on the internet and social media. So what should be done with this information and who really owns it?

Social media is the first port of call when searching for information about a person or more specifically an employee. Applicants nowadays are highly aware that their social networking accounts fall under scrutiny by potential employers. A 2014 CareerBuilder survey found that more half of hiring managers chose not to hire a candidate due to information they found on their social media. The interesting point to note is that many of the reasons cited for not hiring the individual were unrelated to the job they were applying to. (SHRM, 2017) One can easily see a photo on an applicant’s Facebook page and take it out of context. A picture does often say a thousand words, but are those words necessarily truthful. As a result, more and more people, young professionals in particular, have chosen to go by different names on social media. They do not wish to be judged by aspects of their personal life and employers should listen that. Many young professionals are of the mind-set that they are capable of adopting two personas, a professional and personal one. So why is it necessary to judge an employee on aspects of their personal life, when the person who greets them in an interview and subsequently everyday working life is the perfect candidate for the job? Employers need to focus more efforts into the skills and potential their workers can bring to the table and not finding out where they spent their summer in 2012. It is perfectly reasonable and prudent for companies to do background checks on all staff members but when did it become acceptable for organisations to stoop to this level of scrutiny.

In October 2017, a woman named Stephanie was notoriously fired from her job in at Austin Bank Texas in response to viral images containing Stephanie and her fiancé that were posted on social media. The images in question, taken by Wolf & Rose Photography, feature Stephanie posing topless and were meant as a symbol to promote body positivity among our society. Austin Bank Texas however felt the pictures were inappropriate and swiftly severed all ties with Stephanie. When discussing the incident in an interview, she said she felt like “home life and work life are two totally different things”. Stephanie further goes on to describe herself as “a model employee” who “was moving up fast” in the company. (DiNuzzo, 2017) Was it right for Austin Bank Texas to fire an employee based on a photoshoot which had nothing to do with the role she played in the bank? It is in my opinion unethical to access and use information unrelated to the job in question as a means to gauge staff potential. Not only that but it is also often pointless and a waste of company resources.

Is the use of information about staff related to job at hand and gathered during working hours always ethical? In my opinion, whilst the information is the property of the company and they are well within their rights to study that information, there is often too high an emphasis placed on the insights it brings. If a company were to rank its software engineers by who completed the most complex tasks in the shortest amount of time then they would be incorrectly placing too high an emphasis on a set of numbers without considering the factors which have contributed. The programmer at the top of the list may have closed themselves off from fellow workers, be incapable of working with others and unable to take direction. Is it fair to say then that this person is the best the company has to offer? The programmer who may have ranked tenth on the list may have taken longer to complete certain tasks as they were busy helping fellow members of staff with their projects. The numbers analysed don’t reflect points such as these and don’t reflect the human element involved in software engineering. Shouldn’t the impact a programmer has on their fellow employees be seen in equal importance as their impact on the company as a whole?

We also should address how competition among software engineers born from an unhealthy emphasis on the information collected related to productivity can have a negative impact on the working environment. If employees feel as though analytics tools such as Hackystat are too invasive and the data they produce is being used to pit fellow programmers against one another then they are more likely to feel unhappy in the workplace. In a highly skilled sector such as technology companies are constantly looking for ways to attract suitable employees. Websites such as Glassdoor provide detailed ratings on the working conditions in various companies and contain reviews written by employees of those companies. If software engineers feel discontent with their current working conditions they have platforms to share these feelings and doing so will hurt the company’s reputations. If a company dehumanizes its staff through increasing focus on analytics they may inadvertently lose some of their shining talent to a competitor with more favourable working conditions. It may be legal to record data about employees from their use of company infrastructure but this level of scrutiny makes staff uncomfortable as we have seen with Hackstat previously. We can therefor conclude that the misuse of this information is not only unethical but in some cases illogical. If in the search to achieve further productivity one succeeds in alienating and driving away their staff then that surely defeats the original purpose.

There is also the issue of ethics when using analytics as a method of decision making. As discussed previously even Google, a company which is a firm believer in the power of technology, agrees that decisions need to be made by people and not algorithms when it comes to the hiring and promoting of staff. For example, if a company used predictive models to decide not to train people who are on the verge of being either fired or given promotions, they are making decisions based on what an algorithm says may happen, as opposed to what employees are actually doing. People are unpredictable by nature and can be influenced by unknown factors.

There are also the legal implications of choosing to fire employees due to an algorithm deeming their performance unsatisfactory. Adopting these techniques will undoubtedly lead to wrongful termination suit and possibly rightly so as these measure are sometimes unfair to employees. In 2011, Google’s People Analytics team used data to find the qualities of the best and worst managers. However they didn’t use the results as a method of getting rid of managers who possessed the "worst" traits. Instead, the People Analytics team used the insights to improve the overall performance of their management team. Data analytics are a wonderful tool to help improve the daily running of the company but things get controversial when they are used as a way to determine the weakest link. (Straz, 2017)

Then there is issue of analysing irrelevant information and forming decisions based to the findings. Employers looking at HR analytics often get lost in the smaller details and focus on the wrong things. On example of this might be that analytics may declare that employees who live closer to the office are less likely to leave than those who live farther away. Should the company then choose not to employ and promote skilled people who live more than a half hour from the office? It’s probably not the wisest decision. While the use of specific data is legal, it is often not ethical. How can employees be fairly evaluated on decisions they haven’t made, but data predicts they may do? When analysing data, one must consider the big picture as opposed to focusing on minute details.

The issue of ethics in a world when technology has led to a significant decrease in privacy is a hot topic currently. The line between what is an acceptable level of scrutiny has become considerably more blurred. While the analysis of software engineers can lead to deeper insights into the process allowing for increased productivity, this constant stream of data collection leaves said engineers feeling uncomfortable. A balance needs to be reached as to what is an acceptable level of information to be collected and studied. Technology has the key to increased productivity but there are social implications in this increased focus of data collection and analysis.

**References**

Akhnoukh, N. (2017). *You CAN (And Should) Measure Software Engineering Performance*. [online] Engineering.kapost.com. Available at: http://engineering.kapost.com/2015/08/you-can-and-should-measure-software-engineering-performance/ [Accessed 14 Nov. 2017].

Codeclimate.com. (2017). *Code Climate Product*. [online] Available at: https://codeclimate.com/product [Accessed 15 Nov. 2017].

Construx.com. (2017). *Measuring Productivity of Individual Programmers*. [online] Available at: http://www.construx.com/10x\_Software\_Development/Measuring\_Productivity\_of\_Individual\_Programmers/ [Accessed 14 Nov. 2017].

Cs.ubc.ca. (2017). *Computational Intelligence and Knowledge*. [online] Available at: https://www.cs.ubc.ca/~poole/ci/ch1.pdf [Accessed 9 Nov. 2017].

DiNuzzo, E. (2017). *A pregnant woman was reportedly fired over her viral body-positive photo shoot*. [online] Thisisinsider.com. Available at: http://www.thisisinsider.com/plus-size-pregnant-woman-fired-for-body-positive-photos-2017-10 [Accessed 20 Nov. 2017].

Johnson, P. (2013). *Searching under the Streetlight for Useful Software Analytics*.

Mathiprakasam, M. (2017). *OracleVoice: The Road To Analytics As A Service*. [online] Forbes. Available at: https://www.forbes.com/sites/oracle/2014/09/26/the-road-to-analytics-as-a-service/#7e37190b3622 [Accessed 17 Nov. 2017].

Nisen, M. (2017). *Google came up with a formula for deciding who gets promoted—here’s what happened*. [online] Quartz. Available at: https://qz.com/299112/google-came-up-with-a-formula-for-deciding-who-gets-promoted-heres-what-happened/ [Accessed 18 Nov. 2017].

Poole, S. (2017). *The 10 Skills Employers Most Want In 2015 Graduates*. [online] Forbes.com. Available at: https://www.forbes.com/sites/susanadams/2014/11/12/the-10-skills-employers-most-want-in-2015-graduates/#6fb5253b2511 [Accessed 8 Nov. 2017].

SHRM. (2017). *POINT/COUNTERPOINT Should Employers Use Social Media to Screen Job Applicants?*. [online] Available at: https://www.shrm.org/hr-today/news/hr-magazine/Pages/1114-social-media-screening.aspx [Accessed 17 Nov. 2017].

Straz, M. (2017). *HR Analytics: How Should Big Data Be Used in the Workplace?*. [online] Entrepreneur. Available at: https://www.entrepreneur.com/article/271753 [Accessed 16 Nov. 2017].

Successstory.com. (2017). *Max Levchin Story - Bio, Facts, Networth, Family, Auto, Home | Famous Founders | SuccessStory*. [online] Available at: https://successstory.com/people/max-levchin [Accessed 14 Nov. 2017].