Finn Jaksland

18325687

CSU33012 - Measuring Software Engineering

Measuring Software Engineering

In this ever-expanding world of technology we are not only forming new ways to form data but also interpret it. In this essay, I will present different ways an employer can collect data, interpret it and the ethics of doing so.

Existing Technologies and Their Potential Usage

There are a wide variety of technologies relevant to this discussion, marketed both towards professional and personal use. I will attempt to cover a broad range of these technologies in order to provide a full perspective into this topic.

At the most basic level, let's first look into why one would use these sorts of technologies for measuring software engineering. Why are they being used? What sort of benefits would they bring to a project? The potential ability of a current technology to crunch and track data far exceeds the ability of any human. A computer, given the right parameters and programming, could potentially give unbiased feedback and provide statistics that could give insight for a software engineering company. A software engineering company could potentially find certain pairings of software engineers that work well together, know what environments a software engineer thrives in or even calculate the scope of future projects based on the time taken to develop similar features in past projects. All this and more can be provided by pre-existing

technology, provided the right data is collected and interpreted in the right way. It's no wonder that this is becoming increasingly more discussed in today's world of software engineering.

Git, a popular open source tool for version control used in services like Github and Bitbucket could be one way to measure data. At its most rudimentary level, Git allows you to maintain several branches of the same project, merging branches once they are ready to be implemented in the same build. Users of Git can create pull requests to implement their own builds of the project, which is ideally peer reviewed and approved. As users commit changes to a project, they can provide documentation in the form of comments. As such, Git is a tool that favours co-operation and communication between software engineers working on collaborative projects. While synonymous with the open source community, Git is used by closed source groups too. Each line of code has its own recorded history, who wrote it and when. The ability to revert back to previous versions of software builds allows developers to undo any damaging changes that can't be easily patched otherwise.

In my opinion, Git can provide a wealth of data that can be interpreted. One might think that an easy measure to go by would be how many lines of code a software engineer writes. However this isn't so simple. Measuring software engineers based on how many lines of code they write has famously been compared to "measuring progress on an airplane by how much it weighs." by Microsoft's Bill Gates. Certain studies have found a rough correlation between lines of code written and number of defects found in said code alongside a correlation between lines of code and overall time taken to complete a project. Incentivising software engineers to create lengthy code rather than incentivising them to write effective code is counter-productive to one

of the core ideas in programming. It's my belief that other features of Git however can be used to measure software engineering. For example, the continued usage of the various aforementioned features of Git, can be in measuring productivity in the form of engagement. One can measure the engagement of a software engineer with a project based on how many different actions they take on a project (commits, merges, pull requests, branching etc). One can also use the software engineers overall data on a source control service to make an educated guess on their strengths, weaknesses and compatibility working with other types of software engineers based on their actions performed on other projects.

Occupeye, a sensor-based solution created by FM:Systems, is a technology created to allow organizations to collect data based on when a workplace is occupied. According to their official website, Occupeye uses a series of motion and heat sensors contained in a device that can be placed underneath a workplace that will detect whenever someone is sitting at that workplace. They claim that this fully integrated system will enable users to collect and interpret workspace utilisation in real time. Essentially, an employer can keep track of when a desk is occupied for and for how long. Occupeye cites this is more so for energy management but I personally believe that a technology like this can be used more effectively to keep track of how long a software engineer wanders from a desk, be it for a fresh cup of coffee or a toilet break. Used in tandem with other technologies which I will go into further detail about, it is entirely possible to keep track of what a software engineer is doing at almost every given moment that they are being paid for. In the same way you might measure engagement by how often a software engineer is

performing actions on your preferred version control service, you can measure engagement based on how often they are where you want them to be.

Fitbits are more widely known as a commercial product however there has been a rise of companies using fitbits to monitor employee health. Fitbits are a line of digital watches that function as fitness devices, measuring daily steps taken, calories burned among other features geared towards improving the health of the user. While not present in most models, GPS is available in the higher range models. Combined with the apps that the Fitbit is designed to use, an employer can continuously monitor the basic health of employees, in work and if they opt to wear, outside working hours too. Naturally this does not directly tie into measuring the quality of a software engineer's work however in software engineering companies, however one can use the data extracted from this to correlate employee stress with certain factors of the workplace and find ways of lowering that stress. Does a software engineer have a higher heart-rate when working on certain projects? Are the employees who walk less in their free time less likely to reach the weekly quota? These are just a few examples of correlations that could not be measured otherwise without the data provided.

Tobii is a company that specializes in eye tracking technology. They sell a variety of glasses, accessories and even software that can collect data on eye movements. This could be as simple as tracking what a user is focusing on on their screen but certain products like the Tobii Pro Glasses 2 possess the ability to capture and store a video feed. This data could potentially be used to find hotspots, areas on the screen where a software engineer is looking and verify what is on that part of the screen. I personally feel this could be used for identifying and eliminating distractions in the workplace. For instance, if a software engineer had non-work related things

visible on their monitor alongside their work, one could determine if they are reading the emails or not. As these products capture their own video too, this could be used to verify when employees are reading something, paying attention to the slides in a meeting or even if there's something in the workplace distracting them from work.

Moving away from more flashy pieces of hardware, I want to briefly cover software in particular. At the time of writing this, the COVID-19 pandemic has caused a massive rise in remote workers, particularly in the tech sector. Software is going to be easier to distribute to each remote worker's home than any piece of hardware which is why I feel important to remark on these in particular.

Sneek is a video conference software with the twist that it has the ability to access your webcam, even when not in a conference. Sneek's video conference software can be configured to run in the background of a computer, recording the web camera the webcam at all times.The advertised intent here is to see that employees are constantly sitting at their computer, even going so far as to generate a composite image of all employee webcam every few minutes (this interval is customisable) so you can monitor at a glance who is sitting at their designated workstation and who is not. Naturally this works similar to the Occupeye technology I mentioned earlier, as it can be used to track how long an employee is at their workstation with the added benefit that this can easily be used with remote workers.

StaffCop is a computer surveillance software that has similar features to Sneek but while Sneek advertises itself as a video conference service, StaffCop is a far more complex employee surveillance software. According to their official user manual, StaffCop along with the audio/video monitoring offers services that include but are not limited to recording keystrokes,

taking screenshots and an alert system that notifies an employer if a certain banned website or application is used. StaffCop offers a comprehensive data analysis software and uses encryption to mask what data it has collected. It remains invisible as it can to the employee, essentially a black box of data collection they have no control over. The employer however has access to view this data unencrypted and fully customise which faucets of data they want recorded and interpret. Most obviously, all these features center around employee engagement. Is the employee sitting at their desk? What's on their screen? What are they typing and how often are typing? Are they listening to anything in the background while they work? This software allows these sorts of questions to be answered easily, the various ways the data provided by this software is virtually endless.

The Ethics of It All

Ethically, I have some concerns regarding all of this technology. While it's expected that employers would track the work completed by employees and even the things they do outside of office hours to a degree, I feel there needs to be a well defined boundaries. In particular I'll be discussing some of the data processing laws laid out in the European Union's General Data Protection Regulations. Since looking at the technologies covered in this essay alone, I can think of many different ways where this data could be misused by a particularly amoral body. Furthermore I think the overuse of these technologies might only serve to whittle down the trust in an employer/employee relationship.

Global Data Protection Regulation, or GDPR for short, is the data protection law put into effect by the European Union in 2018. It sets a standard for data privacy and data security, that companies must follow or risk fines. While people might be most familiar with how these laws

pertain to consumers, it also regulates data surrounding employees. Under GDPR, employees have a right to information pertaining to the collection and processing of their own personal data. Data controllers are required by law to grant them access to the personal data and supplementary data they hold. They have a right to correct the record on any inaccurate or incomplete data. If an employee considers the collection of said data unlawful or inaccurate, the data controller can be restricted from processing their data. Employees have a right to object to the data being used for marketing, historical or scientific research. Employees also have a right to be given the data on themselves and reuse it how they see fit.

I think these rights are fairly comprehensive yet vague enough to catch any unethical future data-collection technology we currently do not have but I still would like to touch on the ethics of usage of the previously mentioned methods of data collection.

I think that each of these technologies, in isolation, sound good on paper. The majority of them don't seem particularly invasive and can even provide recreational benefits to the software engineer. Access to a source control service and perhaps even data on your usage of it, is a potentially handy tool. Similarly as I mentioned earlier, Fitbits are a very popular device for general consumers. A software engineer might gain some health benefits from using one regularly to monitor daily step count and other things.

These technologies alone don't seem to cross any boundaries, however I feel more reprehensive once we start intersecting these technologies with each other. Consider a hypothetical workplace that employed each of the technologies for measuring software engineering mentioned above. How well does the data synergise with each other, how much can you know about a person simply from using the aforementioned data collection methods and

how might this be interpreted? Consider the hypothetical worker John Smith working in this workplace.

"John Smith wakes up and the motion of walking around will trigger the pedometer of his Fitbit. John's data controller knows what time he woke up and can compare it to all the previous times he woke up. John Smith eats his breakfast and logs it in on the Fitbit Food Tracker. From the GPS system on his Fitbit his employer can see how long it takes him to leave the house and compare that to how long he usually leaves the house. John's data collector knows what food he eats that morning and every other morning.

John Smith arrives at work, he places on his eye-tracking glasses and sits at his desk, triggering the motions sensors placed in it. John's data collector knows when he sat at his desk exactly after entering work. He boots up his PC, triggering the surveillance software and starts looking through his emails. His data collector knows via StaffCop what his inputs are, which applications he has open and the eye tracking can be used alongside that to determine what John Smith is focusing on. A screenshot taken via his webcam, confirms using basic facial recognition software that someone is currently occupying the desk. He writes some code, runs it through test cases and commits it via the version control software. The data controller has data on what he committed.

For lunch John talks to a fellow software engineer Jane Doe. The proximity of the two individuals can be determined via GPS as previously established and can be noted by the data controller. Afterwards John goes back to his work, commits more code to his branch before heading home."

I find that when the data measurement is contextualised in this fashion, it seems far more invasive than before. There is almost nothing significant that the data controller has not collected on John. His actions both outside and inside worked are measured. While this very specific intersection of technology is unlikely and seems bizarre by today's standards, it's entirely within scope given the recent rise in demand for workplace data collection.

This is a lot of data and one has to give it thought how this could possibly all be processed. It's an impossible task for a human to simply manually monitor all this data, even for one person. The usage of interpretation software, proprietary from the data collection technology companies themselves, open-source or simply what is created in-house, is the primary way of interpreting this data. My concerns regarding ethics in this matter is with regards to the potential downsides of this sort of data interpretation. Consider a machine-learning model that is trained on the ideal habits of an ideal software engineer. This ideal but fictional software engineer meets every quota, works consistently, is always engaging with work and only has positive interaction with other software engineers. This model is incentivized to reward employees that best resemble the model of this software engineer based on their data and list when employees deviate from this. In a more hands-free approach, the machine learning model might even be given the power to make executive decisions without the interference of a human, skipping the need to alert anybody in the first place.

At first glance, this sounds efficient, unbiased and in a world where machine learning is a very popular field, not outside the realm of possibility. Yet one of the biggest concerns with the machine learning model is the black box problem. A black box model will design its own algorithm based on the data it is fed and incentivised to use. These algorithms will be designed in

a way that is easily parsable for this machine but will more often be completely incomprehensible to even the creator of this machine learning model. Machines after all "thinks" differently to the way a human might think. What might seem an obvious way of approaching a problem for a human might not be possible for a machine and vice versa. If inadequately trained, this machine learning model has the potential to cause harm that can't be foreseen because we simply cannot determine how the machine makes its decisions.

What if a pregnant worker is deemed a useless worker due to working less hours over an extended period of time, is taking more bathroom breaks than previously recorded and has an above average level of missed work days? A human might otherwise ignore this but a machine that is trained solely on the data we have discussed will not change their judgement nor can we ever fully understand how the machine reached this judgement. It's in this unchecked level of control that a machine learning model for the measurement of software engineering that I personally believe could be very unethical. Furthermore the black box problem makes it more difficult for an employee to understand how their data is being interpreted, which as per GDPR, they have a right to know. How can the complex seemingly random at times behaviour of an ill-trained machine learning model be explained in terms where the software engineer employee knows exactly how their data is being interpreted and collected?

Furthermore excessive data collection could lead to a potential breakdown in trust. There is a studied correlation between high-trust organizations and employee productivity, by reducing that trust one could get the opposite effect of improving productivity.

Conclusion

So in conclusion, it is possible to measure software engineering using current existing technologies. While such technologies are legal by European Union's regulations on data protection for employees, I personally feel that some tread into the territory of being unethical especially when used in conjunction with several different technologies and caution must be practiced in the algorithms formed to interpret this data.

References

Git Documentation - *https://git-scm.com/doc*

Github website - https://github.com/

Occupye website - https://www.occupeye.com/

Tobii website - https://www.tobii.com/

Fitbit website - https://www.fitbit.com/

StaffCop website - *https://www.staffcop.com/*

"Bosses started spying on remote workers. Now they're fighting back" by Alex Christian - https://www.wired.co.uk/article/work-from-home-surveillance-software

"Your company's creeping interest in your body" by Richard Gray - https://www.bbc.com/worklife/article/20171110-your-companys-creeping-interest-in-your-body

"Do you want your company to know how fit you are?" by Emily Young - https://www.bbc.com/news/business-33261116

"Employees at home are being photographed every 5 minutes by an always-on video service to ensure they're actually working — and the service is seeing a rapid expansion since the coronavirus outbreak" by Aaron Holmes - https://www.businessinsider.com/work-from-home-sneek-webcam-picture-5-minutes-monitor-video-2020-3?r=US&IR=T

"The Neuroscience of Trust" by Paul J. Zak - https://hbr.org/2017/01/the-neuroscience-of-trust/

"The 'black box' metaphor in machine learning" by Dallas Card - https://towardsdatascience.com/the-black-box-metaphor-in-machine-learning-4e57a3a1d2b0

"Analysis Of Source Lines Of Code(SLOC) Metric" by Kaushal Bhatt, Vinit Tarey, Pushpraj Patel - *https://www.researchgate.net/publication/281840565_Analysis_Of_Source_Lines_Of_CodeSLOC_Metric*

"An Investigation of the Relationships between Lines of Code and Defects" by Hongyu Zhang - https://www.researchgate.net/publication/316922118_An_Investigation_of_the_Relationships_between_Lines_of_Code_and_Defects

"The Correlation among Software Complexity Metrics with Case Study" by Yahya Tashtoush, Mohammed Al-Maolegi, Bassam Arkok - https://arxiv.org/ftp/arxiv/papers/1408/1408.4523.pdf

"A SLOC Counting Standard" by Vu Nguyen, Sophia Deeds-Rubin, Thomas Tan, B. Boehm - *https://www.semanticscholar.org/paper/A-SLOC-Counting-Standard-Nguyen-Deeds-Rubin/1110f97bdbd475ec4b3abdd59e5147583bd50332*

"Some misconceptions about lines of code" by J. Rosenberg - *https://ieeexplore.ieee.org/document/637174*

"Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey" by Vanessa Buhrmester, David Münch, Michael Arens - https://arxiv.org/abs/1911.12116

"Towards recognizing and rewarding efficient developer work patterns" by WIll Snipes, Vinay Augustine, Anil R. Nair, Emerson Murphy-Hill - https://people.engr.ncsu.edu/ermurph3/papers/icse-nier13.pdf

General Data Protection Regulation - https://gdpr-info.eu/