Measuring a Software Engineer

By Claire Adams
Software Engineering
Trinity College Dublin
Fall 2019

Introduction

What is software engineering?

Software engineering is the process of taking "inexact problems" and creating "exact solutions" to satisfy users.[1] Often the most challenging problems software engineers face come from having to figure out how to translate poorly-specified desires into a product or feature that the user actually wants.[2]

What is software engineering productivity broadly?

The actual metric or metrics used to quantify software engineering productivity vary widely and are the main topic of the rest of this paper. However, even without discussing the concrete ways in which employers and engineers measure productivity, it is important to agree on a general understanding of what it means to be productive.

In economics and businesses generally, "productivity is the ratio between the amount of goods or services produced and the labour or expense that goes into producing them."[3] Since software engineering is the process of creating products to satisfy users, software engineering productivity must broadly be based on the ratio between the satisfaction levels of the user and the labour or expense that went into producing the product.

Why is measuring software engineering productivity important?

In many software engineering companies, as in many companies in general, it is logical that managers and executives want to get as much value out of each employee as they can to help keep their profit margins as high as they can. However, it often takes many months of teams of software engineers working together to develop a product to give to users.[4] Thus, it is clearly difficult to ascertain how useful or productive one engineer is in comparison with others; even once a company has a product for users and can gauge user satisfaction, it is

[1] https://nortal.com/blog/the-myth-of-developer-productivity/
[2] Stephen Barrett, Software Engineering Module, Trinity College Dublin, Lecture 8: Requirements Engineering.
[3] http://www.umsl.edu/~sauterv/analysis/488_f02_papers/SoftwareProductivity.html
[4] Stephen Barrett, Software Engineering Module, Trinity College Dublin, Lecture 5: Agile Methods.

hard to know how much impact one engineer had on that product. The field of software engineering productivity research appears to have appeared out of this struggle to quantify the impact of individuals.

Additionally, the purpose of measuring software engineering productivity also is to "support managerial decision-making during software development and testing."[5]

Measurable Data

In discussions about software engineering productivity data, there appear to be five main types of measurable data that can give employees a sense of how productive their engineers are. These types are code-quality metrics, time-driven metrics, process-driven metrics, workload metrics, and metrics from wearable technologies.

Code-Quality Metrics

Code quality metrics are based around the idea that "producing numbers that characterise properties of code" will help identify productive engineers.[6] Some of these techniques include tracking:

- Lines of code (LOC) - count the lines of code each engineer writes[7]
- Number of defects/bugs created[8]
- Number of bugs closed[9]
- Number of commits
- Number of comments on pull requests
- Number of lines of code with test coverage

Time-Driven Metrics

Time driven metrics focus on identifying how quickly software engineers work and how accurately they can predict how long projects will take. This metric can also be used to identify employees who are not working. Some of the techniques to gather this type of data involve tracking:

- Number of hours worked
- Accuracy of time estimation
- Time before delivery - the time from the business request to the completed system being in production
- Time in progress - time from when work is started on an item until the item is delivered
- Time in phase - how long the project is in each phase (design phase, development phase, quality assurance phase, code review phase, deployment phase)[10]

---

[5] Fenton, Norman E, and Martin Neil. "Software Metrics: Successes, Failures and New Directions." *Journal of Systems and Software*, vol. 47, no. 2-3, 1999, pp. 149–157., doi:10.1016/s0164-1212(99)00035-7.
[6] "Software Metrics: Successes, Failures and New Directions."
[7] "Software Metrics: Successes, Failures and New Directions."
[8] "Software Metrics: Successes, Failures and New Directions."
[9] "Software Metrics: Successes, Failures and New Directions."
[10] https://nortal.com/blog/the-myth-of-developer-productivity/

- Responsiveness to emails/Slack messages
- Responsiveness to pull requests

Process-Driven Metrics

Process-driven metrics are often derived from specific software engineering methodologies, such as agile and kanban. Agile methodology is a process by which teams develop software in iterative cycles.[11] Kanban is a scheduling system to "limit the buildup of excess inventory at any point in production" and reduce overworking.[12] Some of the techniques to gather this type of data involve tracking:

- Number of Story points engineers take on in a sprint[13] (Story points are part of the Agile process where one estimates how long it will take to implement a small feature or fix a bug.[14] A sprint is "one timeboxed iteration of a continuous development cycle."[15])
- Number of Story points engineers complete in a sprint
- The ratio of completed Story points to the number of Story points taken on
- Number of items worked on at a time in the Kanban process, which encourages developers to see projects to completion rather than work on multiple projects at once[16]

Workload Metrics

Workload metrics try to help employers understand how much work software engineers have and how they are managing their time. Some of the techniques to gather this type of data involve tracking:

- Number of emails/Slack messages received per day[17]
- Number of overtime hours worked[18]
- Number of days coding per week
- Amount of sick leave taken

Wearable Technology Metrics

Wearable technology metrics focus on tracking the locations, conversations, and sometimes even heart rate and other health factors.[19] Some of the techniques to gather this type of data involve tracking and recording:

- Proximity to others[20]
- Number of conversations with given people[21]

---

[11] https://en.wikipedia.org/wiki/Agile_software_development#Iterative,_incremental_and_evolutionary
[12] https://en.wikipedia.org/wiki/Kanban
[13] https://nortal.com/blog/the-myth-of-developer-productivity/
[14] https://www.visual-paradigm.com/scrum/what-is-story-point-in-agile/
[15] https://yodiz.com/help/what-is-sprint/
[16] https://nortal.com/blog/the-myth-of-developer-productivity/
[17] https://statustoday.com/
[18] https://statustoday.com/
[19] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf
[20] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf
[21] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf

- Conversations with specific people[22]
- Location within offices[23]
- Heart rate[24]

Computational Platforms for Measurement

There are various pre-existing platforms that employers can use to track their employees. Platforms that focus on measuring software engineering data, in particular, have been popping up in recent years.[25] Employers who wish to learn more about their engineers can also use platforms that are not software engineering specific.

Software Engineering Specific Platforms

GitPrime - Analytics to Understand Bottlenecks and Communication[26]

GitPrime is a platform designed to get information from git repos “to make engineering teams more successful.” It identifies bottlenecks, compares work trends, and keeps “a pulse on the health of your software teams.” This product measures:

- Number of days a software engineer codes
- Number of commits per day
- Impact of a commit - “The severity of edits to the code base vs repo.”
- Efficiency - “The ratio of churned code to contributed code.”
- Disagreements in pull requests
- Time to resolution for pull requests
- Number of reviewers per pull request
- Responsiveness of reviewers to pull request comments

CodeClimate - Analytics to Understand Pull Requests[27]

CodeClimate is a platform to get information from git repos “to quickly and accurately diagnose where work gets stuck, who needs help, and where the greatest opportunities lie for continuous improvement.” This product measures:

- Number of pull requests in each state (pushed, open, reviewed, approved, merged, closed)
- Pushes per day
- Pull request size
- Time to open a pull request
- Time to respond to pull request comments
- How many pull requests are approved without comments
- Average number of lines of code per pull request

---

[22] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf
[23] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf
[24] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf
[25] https://www.gitprime.com/company/
[26] https://www.gitprime.com/
[27] https://codeclimate.com/

Hackystat - Open Source PSP Analytics Platform[28]

Hackystat is "an open source framework for collection, analysis, visualization, interpretation, annotation, and dissemination of software development process and product data."[29] Hackystat was created to automate the collection of software developer data for developers using the "Personal Software Process (PSP). The primary goals of the PSP are to improve project estimation and quality assurance" for individual developers.[30] Hackystat works by attaching sensors to development tools, which then automatically send data to be analyzed.[31] This product measures:

- What files are being modified
- How many lines of code are being written
- How many new classes are being written
- How many new methods/functions are being written
- How often test cases pass or fail
- How often test cases are being run

Code Time - Open Source Time-tracking Plugin for VS Code[32]

Code Time is an open source time-tracking plugin for VS code that provides data for individuals for 90 days at a time. All data is anonymized to protect users' privacy. This platform is free. This product measures:

- Number of hours coded per week
- Number of lines of code added per week
- Number of lines of code deleted per week
- Number of characters added per week
- Number of characters deleted per week
- Total number of keystrokes
- Most active times of the day and of the week
- Percentage of code written in each language

Wakatime - Open Source Time-tracking Plugin[33]

Wakatime is an open source platform to let software engineers "know exactly how long [they] spend coding." This platform measures:

- What programming languages are used the most
- How much time one spends coding per commit
- How much time one has spent coding in a given programming language

---

[28] https://hackystat.github.io/
[29] https://hackystat.github.io/
[30] Johnson, P.m., et al. "Beyond the Personal Software Process: Metrics Collection and Analysis for the Differently Disciplined." *25th International Conference on Software Engineering, 2003. Proceedings.*, 2003, doi:10.1109/icse.2003.1201249.
[31] https://hackystat.github.io/
[32] https://marketplace.visualstudio.com/items?itemName=softwaredotcom.swdc-vscode
[33] https://wakatime.com/features

Gamification of Version Control[34]

In one undergraduate university class, researchers used gamification of version control to encourage students to commit more frequently, which they claimed was a good software engineering practice. This experiment had a board showing which students had the most commits. This experiment is one example of a way to measure the number of commits and then use competition to change the behavior of software engineers. This approach measures:

- Number of commits
- Changes in the number of commits over time

General Employee Monitoring Platforms

Isaak by StatusToday - Analytics on Overworking and Connectivity[35]

Isaak is a StatusToday product that "provides real-time wellbeing insights including email overload, overworking signals and focus hours." It uses "people analytics to understand email responsiveness, work completed outside of normal office hours and time spent without distractions." The goal of this product is to prevent employee burn-out and identify people who are at risk of leaving. This product measures:

- How frequently people are working overtime and at what times are they working
- How many emails people get and if they are overloaded
- Tracks which employees are connected to each other and which consumers are connected to employees

Humanyze - Employee Badge Tracking[36]

Humanyze uses "KPI's, surveys, or sensors" to "uncover patterns on how and where teams work" to "draw correlations between key contexts, validate interventions, and reveal impacts or changes over time." It uses metadata to draw conclusions instead of using data about specific employees. This product measures:

- "Communication patterns" from Slack, email, Skype, and Office 365[37]
- Employee location in an office[38]
- Conversations between employees[39]

Algorithmic Approaches to Analyzing Productivity Data

Once a company amasses software engineering data, they must analyze it so that it can provide useful information to individual developers and managers. Managers may wish to

---

[34] Singer, Leif, and Kurt Schneider. "It Was a Bit of a Race: Gamification of Version Control." *2012 Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS)*, 2012, doi:10.1109/gas.2012.6225927.
[35] https://statustoday.com/
[36] https://www.humanyze.com/
[37] https://www.humanyze.com/
[38] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf
[39] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf

use this data to make decisions about promotions and the retention of their software engineers.

This section covers several Artificial Intelligence (AI) based algorithms, a general overview of data mining, and a high-level discussion of the Personal Software Process technique. When analyzing software engineering productivity data, companies most likely use AI-based algorithms. Within these algorithms, there are two main types: “fully automatic,” which have no human involvement in the decision-making process, and “partly automatic,” where humans use an algorithm to help them make a decision, but the algorithm does not decide for them.[40] Additionally, techniques such as data mining can help extract patterns from data. Finally, when considering software engineering data, in particular, one common analysis method is the Personal Software Process methodology, which allows software engineers to gain insights about how to optimize their own workflows.

AI Algorithms and Models

Neural Networks

Neural networks are distributed systems that “learn” to make decisions about unlabeled data based on labeled examples.[41] They are designed to recognize patterns and depend on a fixed set of labels. Neural networks can be used in a fully automated system since, once they are trained on labeled data, they are designed to classify more data without the input of humans. Thus, they can be used to classify whether or not employees are meeting expectations, for example, much like they are used to classify spam email messages. The limitations of neural networks include the fact that there must be a predefined set of labels that all data will be matched with and the fact that the quality of the algorithmic matching is heavily dependant on the quality of the labeled training set; that is to say, a company must have a lot of data on their employees and must already have labeled their employees as good verses bad hires, or what have you, before the neural network can make quality decisions.[42] Any biases in the data set will be reflected in the results from the neural network.[43]

Fuzzy Logic

The purpose of fuzzy logic is to provide a “mathematical means of representing vagueness and imprecise information” based on the idea that “people make decisions based on imprecise and non-numerical information” all the time.[44] The idea behind fuzzy logic is that it moves away from making binary decisions and instead assigns a weight to different decisions. This algorithm “attaches numeric values between 0 and 1 to each proposition in

---

[40] Borgesius, Frederik Zuiderveen. “Discrimination, Artificial Intelligence, and Algorithmic Decision-Making.” 2018, https://rm.coe.int/discrimination-artificial-intelligence-and-algorithmic-decision-making/1680925d73.

[41] https://cis.ieee.org/about/what-is-ci

[42] https://skymind.ai/wiki/neural-network

[43] https://towardsdatascience.com/https-medium-com-mauriziosantamicone-is-artificial-intelligence-racist-66ea8f67c7de

[44] https://en.wikipedia.org/wiki/Fuzzy_logic

order to represent uncertainty." In making employment decisions such as hiring or firing people, fuzzy logic can help capture the vague qualities an employer is looking for, such as if an employee is competent or not. However, like neural networks, systems based on fuzzy logic can encode biases based on the cut offs for labels provided to the algorithm.[45]

Evolutionary Computation

Evolutionary computation is a process to "come up with a variety of strategies" and then test them against a "predictor, which would act as a surrogate for the real world." Through generating models and then testing their ability to predict events, it is possible to then create new models that are better suited to solving the problems at hand,[46] much how evolution works in the natural environment.[47] Evolutionary computation works well with less historical data than is needed for fuzzy logic or other such approaches. Needing less historical data is good for companies just starting to measure their employees. However, evolutionary computation is also slow at giving results so companies must be patient. Cognizant, an artificial intelligence company, offers a product that uses evolutionary computation to help companies figure out how to optimize processes, such as the "decision cycle" for "responding to software development and testing requests for proposal."[48]

Influence Modeling

Influence modeling is a machine learning model that uses "independent time series [data] to estimate how much the state of one actor affects the state of another actor in the system." It is used to understand how different people interact with and influence each other. Influence modeling works by using individual observations to predict how others will react to one person's actions where actions are defined as changes in state, for example, going from talking to not talking.[49] This model then works well for companies who want to understand who the influential people are in their company.

Data mining

Data mining "provides the techniques to analyze and extract novel, interesting patterns from data."[50] The purpose of applying data mining techniques to software engineering data is that it "can assist software engineers in predicting, planning, and understanding various aspects of a project so that they can more efficiently support future development and project management activities."[51] Data mining can be used to "help software engineers to predict software failures, extract and classify common bugs, identify relations among classes in a

---

[45] https://theconversation.com/to-stop-the-machines-taking-over-we-need-to-think-about-fuzzy-logic-37961

[46] https://www.cognizant.com/ai/blog/informing-business-decision

[47] https://en.wikipedia.org/wiki/Evolutionary_computation

[48] https://www.cognizant.com/ai/blog/informing-business-decision

[49] Pan, Wei & Cebrian, Manuel & Dong, Wen & Kim, Taemie & Fowler, James & Pentland, Alex. (2010). "Modeling Dynamical Influence in Human Interaction Patterns." *Computing Research Repository - CORR*.

[50] Halkidi, Maria & Spinellis, Diomidis & Tsatsaronis, George & Vazirgiannis, Michalis. (2011). "Data Mining in Software Engineering." *Intelligent Data Analysis Journal (IDA).* pp.413-441.

[51] "Data Mining in Software Engineering."

libraries [sic], analyze defect data, discover reused patterns in source code and thus automate the development procedure."[52] Thus, data mining of source code can help engineers be more productive. There are five main data mining components: clustering using unsupervised machine learning techniques, classification using supervised machine learning techniques, frequent pattern mining and association rules, data characterization and summarization, and change and deviation detection.[53]

In general, clustering is used to identify "interesting distributions and patterns in the underlying data." For software engineering in particular, "clustering can be used to define groups of similar modules based on the number of modifications and cyclomatic number metrics" in a programmer's code.[54]

Classification involves assigning data items to predefine sets of classes, often through the use of decision trees. In software engineering, these decision trees can help identify "risky software modules."[55]

Association rules "reveal underlying 'correlations' between the attributes in the data set."[56]

Data characterization "is the summarization of the data characteristics of a specific class of data" that can then be used to find parts of code that satisfy user-specified requirements.[57]

Finally, change and deviation detection finds "the most significant changes in the data" to find, for example, source code changes.[58]

Overall, data mining can be used to extract code quality metrics when applied to source code. It is a generalizable set of principles that can be applied to other sets of data, such as software engineering data, to find patterns to improve productivity as well.

Personal Software Process

The personal software process is focused on relatively simple analysis of data sets about one engineer rather than on AI algorithms to make decisions and predictions about many engineers. It is a process, and thus an algorithm, for software engineers to bring "discipline to the way they develop software and [track] their predicted and actual development of code." The personal software process focuses more on providing individual developers with data about themselves than on providing employers with decisions or suggestions about the quality of their employees, although the data collected in this method can be used for that as well. This process involves multiple phases where the developer learns how to collect the data needed for the process and then learns from the trends they discover about themselves. The four main types of data for this process include measuring: the size of a

[52] "Data Mining in Software Engineering."
[53] "Data Mining in Software Engineering."
[54] "Data Mining in Software Engineering."
[55] "Data Mining in Software Engineering."
[56] "Data Mining in Software Engineering."
[57] "Data Mining in Software Engineering."
[58] "Data Mining in Software Engineering."

project (possibly in lines of code), the time it takes to complete a task, the number of bugs in the final product, and the expected progression on the project verses the actual progression. By analyzing this data using statistical formulas, engineers can see their areas of weakness and of strength.[59]

Ethical Concerns

In general, ethics is “a system of moral principles” that “affect how people make decisions and lead their lives.”[60] Business ethics focuses on ethical dilemmas in the business environment. It “applies to all aspects of business conduct and is relevant to the conduct of individuals and entire organizations.”[61] In technology, ethics often focuses on “whether the very act of innovation is an ethically right or wrong act.”[62] Monitoring software engineering productivity falls in the crosshairs of both business ethics and the ethics of technology -- it both shapes a business by creating a company culture and shapes the larger world by exemplifying, and possibly creating, tools that can be used to monitor people in other industries. Therefore, as with all applications dependant on data from individuals, when measuring software engineering productivity, it is vital to critically assess the ethical implications of collecting data about software engineers and using that data to determine their ability.

It is essential that employers take action to ensure that the data they collect and the decisions they make based on that data does not create toxic cultures of distrust or competition in their employees. I believe that some key ethical concerns employers should consider when monitoring their employees and then making business decisions off this data are:

- Privacy: Where is the line between professional data, or data that can reasonably be construed as belonging to a company, and personal data?
- Security: Can a company responsibly protect the data it collects about its employees? What are the implications if employee data is leaked?
- Accuracy: Can the company be sure that the collected data accurately represent a software engineer’s ability?
- Knowledge: Should employees be informed that they are being monitored?
- Stereotypes: If a company uses data about software engineers to influence their hiring practices, are the models of productive engineers derived from their data spreading any harmful stereotypes?

Privacy

Drawing the line between what data belongs to a company, and therefore is fair game to analyze, and what is personal information that should not be gathered or used by a company to make decisions about the quality of its employees is difficult. However, drawing this line is essential to not only preserving employees’ trust in a company but also to creating a culture

[59] https://en.wikipedia.org/wiki/Personal_software_process
[60] http://www.bbc.co.uk/ethics/introduction/intro_1.shtml
[61] https://en.wikipedia.org/wiki/Business_ethics
[62] https://en.wikipedia.org/wiki/Ethics_of_technology

that protects an individual's privacy more broadly. Essentially, if the tech companies, which thrive on having lots of data,[63] don't respect the privacy of their employees, how can they be trusted to respect the privacy of their customers?

Some metrics such as lines of code or number of hours worked seem reasonable for companies to track because it is clear how these metrics relate to the output of an employee. However, other metrics, such as all keystrokes an employee makes or locations of employees, seem more ethically dubious because they involve more invasive monitoring that does not clearly relate to the expected work product for employees.

Take tracking keystrokes, for example. Some might argue that everything a software engineer types on their company laptop belongs to the company they work for because it is a company laptop and a software engineer uses keystrokes to do their work. However, I believe it is not ethical to track everything an employee types because tracking everything makes it impossible for an employee to have any privacy while on their computer. If an employee jots down a note on a piece of paper at work, it is possible for the employee to throw the piece of paper away without the employee's company ever knowing about it. With keyboard monitoring, this disposability-- this privacy -- is lost because every action, and every digitized note, can be tracked, read, and possibly used as the basis for firing someone. Thus, while a company may assert that it is attempting to increase productivity by using keyboard tracking to identify employees who are off-topic,[64] keyboard tracking appears to be a prime example of an unethical method of tracking employees. Tracking every keystroke means that an employee has no privacy whatsoever when they are using their company laptop. It is an excessive destruction of privacy, with limited, if any, benefits that could not be achieved in other, less invasive ways.

Likewise, I assert that tracking employee location in an office is an unethical invasion of privacy because it suggests that the employees, and not just their work product, belong to a company. While understanding how employees move about an office space can help a company reorganize to be more efficient, this result can be achieved with short-term help from an organizational consultant and does not necessitate the continued, digital monitoring of employee location.

Overall, I contend that the ethical boundary of tracking employees does not include techniques that inhibit an employee from having much, if any, privacy.

Security

Another ethical issue companies need to consider when collecting data on their employees is the company's ability to keep this data secure and what the ramifications will be if this data is hacked. According to one security firm, "In 2018 hackers stole half a billion personal

---

63 http://theconversation.com/tech-companies-collect-our-data-every-day-but-even-the-biggest-datasets-cant-solve-social-issues-118133

64 Yerby, Johnathan. (2013). "Legal and ethical issues of employee monitoring." *Online Journal of Applied Knowledge Management.* pp.44-55.

records."[65] Additionally, from the cycle of news about data breaches, it is clear that even very successful companies struggle to protect private data.[66] Thus, when a company goes down the route of collecting data about its employees, I contend that it has an ethical responsibility to keep this data secure. If a company cannot do so, it should not be collecting data on its employees. Keeping data about employees secure is essential because the leakage of metrics, especially if they are inaccurate or bias, that depict an employee in a negative light could jeopardize the employee's ability to get a job in the future.

Accuracy

If the data doesn't capture the full picture

With monitoring software engineering productivity, in particular, it is important to recognize that often the data collected cannot accurately depict the reasoning behind why an employee might be doing something that is deemed unproductive at a given time. For example, when monitoring whether or not a person is working based on their keystrokes or response times to emails, pull requests, or instant messages, "the monitoring programs cannot know when an employee has an upset stomach and needs to be away from their desk - it just senses that the employee is not currently working. The employers get, in a sense, biased and incomplete data."[67] Unless the employee is constantly monitored in all manners, as in with software, tracking devices, and recording devices, at work and in their private life, which would be a clear and unethical invasion of privacy, it is impossible for the data collected by a company to display the full picture of how and why an employee is performing in a given way. Thus, it is especially unethical for a company to use fully automated AI algorithms to take in data and make decisions about the future of an employee. Employees are human beings, not machines, so they shouldn't be treated as such. Even partially automated algorithms can lead to managers making unfair decisions about their employees. Managers may not fully understand the limitations and possible biases within the algorithms and as a result, may exhibit "automation bias" where they trust the computer's result without analyzing it.[68]

If the data captures a falsely full set of data

Since most software engineers frequently work with data sets, it is reasonable to assume that most software engineers know how to manipulate data to achieve a desired result. For example, if an employee discovers that the number of lines of code they write is directly tied to the likelihood of them getting a promotion, then they can write more lines of code to make them appear more productive when, in reality, they are writing bad code that will most likely make others have to do more work later. This same data manipulation tactic can be applied to the number of hours employees work, the number of bugs closed, and other metrics that are dependant on quantity and not quality discussed previously (see "Measurable Data").

---

[65] https://www.cybintsolutions.com/cyber-security-facts-stats/

[66] https://www.telegraph.co.uk/technology/2019/04/03/millions-facebook-user-records-exposed-data-breach/

[67] "Legal and ethical issues of employee monitoring."

[68] "Discrimination, artificial intelligence, and algorithmic decision-making."

While some might view this accuracy problem as a sign that the perfect metric to measure software engineering productivity has yet to be found, as explained earlier (see “Why is measuring software engineering productivity important?”), it is incredibly difficult to quantify the quality of a software engineer; hence, this paper and the various tools discussed above (see “Computational Platforms for Measurement”) attempt to address this dilemma based on measurable data, which is often easy to manipulate.

Additionally, monitoring employees can decrease employee productivity. It can lead to increased stress in employees and can create distrust and animosity not only between employees and employers but also between other employees by leading to increased competition. In “Legal and ethical issues of employee monitoring”, Johnathan Yerby writes that “monitoring can create a hostile workplace, possibly eliminating the whole point of monitoring in the first place (i.e., to increase efficiency).”[69] Thus, even if a company believes that their metrics and analysis accurately represents the current productivity of their employees, this data may not accurately reflect the potential productivity of an employee if that employee was not subjected to the added stress of feeling as if they were constantly being watched.

Overall, I contend that it is unethical to solely, or even largely, based continued employment decisions on data collected from employees because it is unlikely that the data is accurate. For the same reason, I believe that it is especially unethical to use fully automated systems that make decisions about employees without human input.

Knowledge

I assert that companies must inform their employees about what data they are collecting and how they will use the data. Keeping employees in the loop doesn’t simply prevent the erosion of trust between employees and employers; it also sets a larger precedent where people have control over their data. Initiatives such as GDPR are trying to force companies to be open about what data they track and make them get consent before doing so. However, I believe that many people still don’t know what they are agreeing to when they click “Accept Cookies,” and, more importantly, it is often impossible for a customer to use an application without giving consent.[70] Since employee monitoring is an area where it seems as if it would be difficult for an employee to opt-out of being tracked, companies must seriously consider the ethical ramifications, within their companies and outside of them, of forcing people to hand over various types of personal data in exchange for employment. That is, what kind of data an employer expects to receive from its employees will contribute to a larger, cultural understanding of what is personal data and who has the right to privacy.

Perpetuating Stereotypes

It seems natural to imagine that a company would use the data it collects about its software engineers to create a profile for what they believe to be the characteristics that productive engineers display. Then, this company could focus on hiring people who fit that model.

---

[69] “Legal and ethical issues of employee monitoring.”
[70] https://www.ft.com/content/624f813e-5f5e-11e8-9334-2218e7146b04

Clearly, such a tactic could result in the perpetuation of stereotypes that discriminate against different groups of people because they do not match preconceived notions of what a software engineer looks like or what background they should have. These notions may be further perpetuated by the use of data sets derived from a sampling of people in which there was little to no representation of these minorities.

Even within discussions about measuring software engineering in general there is an idea, and some research, suggesting that some software engineers are 10x more productive than other software engineers, given that they all have the same number of years of experience.[71] Some people interpret this statistic as suggesting that there is *an innate quality* -- just one -- that makes some software engineers more productive than others. This idea that there is a *single* quality that allows an engineer to be 10x more productive that their peers makes it easy to imagine that some companies could, or possibly already do, create a personality profile of a stereotypical engineer that a company believes will be 10x to try and discover the single, yet indescribable, quality that will make their hires more worth their money.[72] Using this template, a company might then hire people who match the stereotype, thus perpetuating it. This threat of spreading stereotypes is particularly harmful in tech fields because of the existing lack of diversity in tech. (In 2018, Google's employees were 30.9% female, 2.5% Black, 3.6% Lainx, and 0.3% Native American.[73]) Since there is a lack of diversity in tech, it is highly probable that any conclusions about what makes a good software engineer will be founded in data that does not fully represent the diversity of skill sets and different types of productivity that minorities bring to the table. Thus, hiring based on models derived from current software engineering productivity data is unethical if companies do not carefully and thoughtfully address the biases they have internalized in their cultures and in their data sets.

Conclusion

Overall, as companies continue to investigate ways to optimize their investments in software engineers and as platforms such as GitPrime and CodeClimate crop up, I believe that it is increasingly likely that I will be monitored at work. Given the wide range of measurable attributes discussed earlier (see "Measurable Data") combined with the current lack of one measurable quality that clearly distinguishes highly productive engineers from others, I will probably be measured not just based on my code quality, but also based on my communication skills, the time it takes me to complete tasks, and possibly even based on who I interact within an office. Since the ethics surrounding employee monitoring, in my mind, are a little murky, and there are few legal regulations about the matter, my hope is that the companies I choose to work for will be open to discussions about the implications of tracking software engineers, not just for the individuals involved but also for the more global precedents such tracking sets.

---

[71] https://softwareengineering.stackexchange.com/questions/179616/a-good-programmer-can-be-as-10x-times-more-productive-than-a-mediocre-one

[72] https://www.quora.com/What-do-10x-software-developers-understand-that-other-programmers-dont

[73] https://static.googleusercontent.com/media/diversity.google/en//static/pdf/Google_Diversity_annual_report_2018.pdf

Bibliography

Journals

- Fenton, Norman E, and Martin Neil. “Software Metrics: Successes, Failures and New Directions.” *Journal of Systems and Software*, vol. 47, no. 2-3, 1999, pp. 149–157., doi:10.1016/s0164-1212(99)00035-7.
- Johnson, P.m., et al. “Beyond the Personal Software Process: Metrics Collection and Analysis for the Differently Disciplined.” *25th International Conference on Software Engineering, 2003. Proceedings.*, 2003, doi:10.1109/icse.2003.1201249.
- Singer, Leif, and Kurt Schneider. “It Was a Bit of a Race: Gamification of Version Control.” *2012 Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS)*, 2012, doi:10.1109/gas.2012.6225927.
- Borgesius, Frederik Zuiderveen. “Discrimination, Artificial Intelligence, and Algorithmic Decision-Making.” 2018, https://rm.coe.int/discrimination-artificial-intelligence-and-algorithmic-decision-making/1680925d73.
- Pan, Wei & Cebrian, Manuel & Dong, Wen & Kim, Taemie & Fowler, James & Pentland, Alex. (2010). “Modeling Dynamical Influence in Human Interaction Patterns.” *Computing Research Repository - CORR*.
- Halkidi, Maria & Spinellis, Diomidis & Tsatsaronis, George & Vazirgiannis, Michalis. (2011). “Data Mining in Software Engineering.” I*ntelligent Data Analysis Journal (IDA).* pp.413-441.
- Yerby, Johnathan. (2013). “Legal and ethical issues of employee monitoring.” *Online Journal of Applied Knowledge Management.* pp.44-55.

Websites

- https://www.quora.com/What-do-10x-software-developers-understand-that-other-programmers-dont
- https://softwareengineering.stackexchange.com/questions/179616/a-good-programmer-can-be-as-10x-times-more-productive-than-a-mediocre-one
- https://nortal.com/blog/the-myth-of-developer-productivity/
- https://static.googleusercontent.com/media/diversity.google/en//static/pdf/Google_Diversity_annual_report_2018.pdf
- http://www.umsl.edu/~sauterv/analysis/488_f02_papers/SoftwareProductivity.html
- https://nortal.com/blog/the-myth-of-developer-productivity/
- https://en.wikipedia.org/wiki/Agile_software_development#Iterative,_incremental_and_evolutionary
- https://en.wikipedia.org/wiki/Kanban
- https://www.visual-paradigm.com/scrum/what-is-story-point-in-agile/
- https://yodiz.com/help/what-is-sprint/
- https://statustoday.com/
- https://www.gitprime.com/company/
- https://www.gitprime.com/
- https://codeclimate.com/
- https://hackystat.github.io/
- https://marketplace.visualstudio.com/items?itemName=softwaredotcom.swdc-vscode
- https://wakatime.com/features
- https://www.humanyze.com/
- https://cis.ieee.org/about/what-is-ci
- https://skymind.ai/wiki/neural-network
- https://towardsdatascience.com/https-medium-com-mauriziosantamicone-is-artificial-intelligence-racist-66ea8f67c7de
- https://en.wikipedia.org/wiki/Fuzzy_logic

- https://theconversation.com/to-stop-the-machines-taking-over-we-need-to-think-about-fuzzy-logic-37961
- http://theconversation.com/tech-companies-collect-our-data-every-day-but-even-the-biggest-datasets-cant-solve-social-issues-118133
- https://www.cognizant.com/ai/blog/informing-business-decision
- https://en.wikipedia.org/wiki/Evolutionary_computation
- https://en.wikipedia.org/wiki/Personal_software_process
- http://www.bbc.co.uk/ethics/introduction/intro_1.shtml
- https://en.wikipedia.org/wiki/Business_ethics
- https://en.wikipedia.org/wiki/Ethics_of_technology
- https://www.cybintsolutions.com/cyber-security-facts-stats/
- https://www.telegraph.co.uk/technology/2019/04/03/millions-facebook-user-records-exposed-data-breach/
- https://www.ft.com/content/624f813e-5f5e-11e8-9334-2218e7146b04

Lectures

- Stephen Barrett, Software Engineering Module, Trinity College Dublin, Fall 2019, Lecture 5: Agile Methods.
- Stephen Barrett, Software Engineering Module, Trinity College Dublin, Fall 2019, Lecture 8: Requirements Engineering.
- Stephen Barrett, Module slides, Trinity College Dublin, Fall 2019, https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/measurement.pdf