Hayden Ivatts

Software Engineering

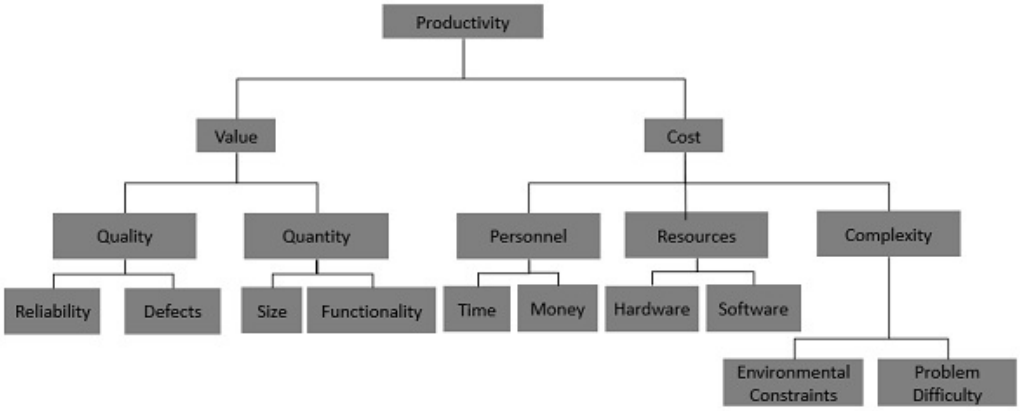
CSU33012- Professor Barrett

5 November 2019

Measuring Software Engineering Report

Introduction

The most fundamental question software engineers like to answer when talking about measuring themselves is “How good is the work I did”. That question is ultra-simplified compared to the complex problems that arise from judging how well a particular engineer or project performs on the job specified. The graph below illustrates some of the major factors that go into deciding if someone works productively or not. However, there are many more nuances and complications that must, and will, be considered when analysing an engineer.



https://www.tutorialspoint.com/software_quality_management/software_quality_measurement_metrics.htm

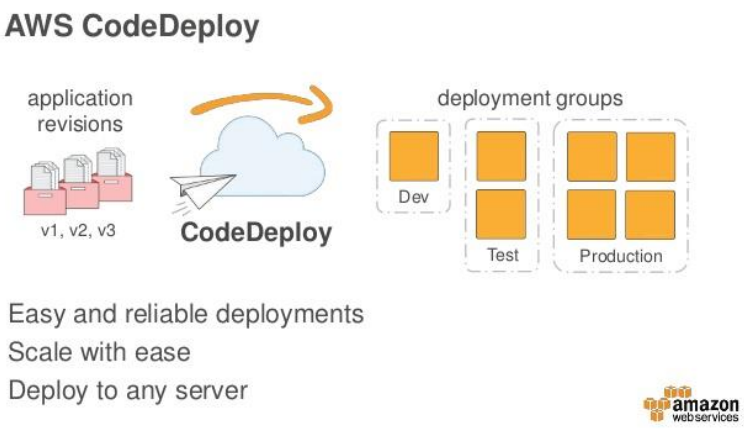
To really answer this vague question engineers must first break down what can be measured. The “what” can then be broken down into three categories: Product metrics, project metrics, and process metrics.

The most obvious measurable of software development stems from the finished product itself. Gathering metrics from finished products result in data about how large the project is, how complex the project is, the number of design features, performance level, and quality. While some of these aspects, such as size or complexity, are simple to measure and understand, many of the features involve very subjective facets that must be examined carefully. For example, quality and performance both require benchmarks to receive a grade or some other type of qualitative data point.

Categories of Productivity Metrics

First, looking at quality as a subjective measurement, there are several pillars that most engineers strive for these include reliability, performance efficiency, security, and maintainability. The two most subjective and most important ones are reliability and security. "Reliability is the risk of software failure and the stability of a program when exposed to unexpected conditions. Reliable software has minimal downtime, good data integrity, and no errors that directly affect user…Security assesses how well an application protects information against the risk of software breaches. The quantity and severity of vulnerabilities found in a software system are indicators of its security level. Poor coding and architectural weaknesses often lead to software vulnerabilities"[1]. While the website cited gives very tidy definitions of how to measure these essential quality factors, it is important to recognize the oversimplification of these metrics. Considerations regarding reliability such as how software responds in extreme user situations, how the software performs with subpar network conditions, and how the software deals with users that are using it much more rigorously then intended. Then regarding security, how does medical software hold up when people's lives are at risk, nuclear software's ability to work properly even in a meltdown scenario, and how would software respond to entirely new forms of computation such as quantum. Although these situations rarely happen to most software projects, they represent an important aspect of how we measure productivity because in many systems the engineer who develops the security patch for a very common situation is given as much credit as the one who develops the patch to a flaw that has a .000001% chance of happening. The oversimplification occurring within product metrics continues, perhaps amplified, in the other categories of metrics.

The next, often overlooked, category of metrics attempts to measure how well the engineer deals with the process of developing and maintain the software. Process metrics include effectiveness of defect removal during development, the pattern of testing defect arrival, response time of fixing bugs when product is out, maintaining product efficiently, and speed of releasing new code to the ongoing product.[2] Process metrics generally measure how well an engineer improves their own work and how consistently they introduce new features or designs to a product. The rise of technology and our cultural obsession with innovation has made process metrics especially fragile because they sometimes involve releasing new code without a real purpose or significant improvement to the original product. For example, "Amazon unfolds new software for production through its Apollo deployment service every 11.7

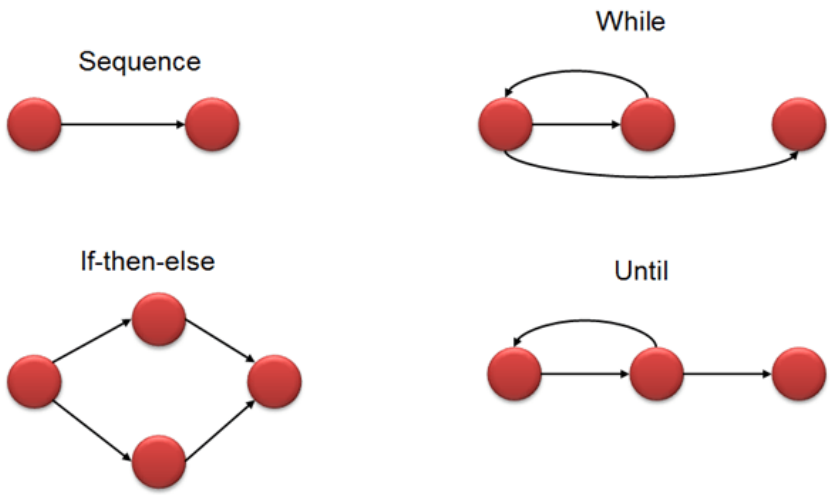


[1] https://www.altexsoft.com/blog/engineering/what-software-quality-really-is-and-the-metrics-you-can-use-to-measure-it/

[2] https://www.tutorialspoint.com/software_quality_management/software_quality_measurement_metrics.htm

seconds"[3]. Amazon uses their abundance of new code as example of innovation and evidence of strong engineering processes. As seen in the image to the left, Amazon takes great pride in their efficient code deployment and even uses it in their advertising for AWS. While updating and releasing new code to your products is essential, measuring productivity should not solely be based off of how many commits Amazon makes to its products. Setting this standard has the potential to reward companies for being innovative who are simply making marginal changes constantly, that may not be necessary altogether. There certainly is some evidence supporting extremely high release rates, "High-performing IT organizations experience 60X fewer failures and recover from failure 168X faster than their lower-performing peers. They also deploy 30X more frequently with 200X shorter lead times"[4]. These statistics attempt make a simple point. More software releases mean better software. While looking at numbers like this is important and undoubtedly tells part of the story of quality software, it is dangerous to look at these numbers and immediately conclude the engineers working on the project are doing a good job.

The numbers mentioned above are only a part of the picture assessing how a project stacks up in terms of process metrics. Maintainability represents a great example of an additional metric that must be added on top of software refreshing to get the full picture on the process of a project. Some engineers use an oversimplified way of categorizing and counting lines of code to measure complexity. They put each line of code in one of four categories which are, physical lines, logical lines, comment lines, and non-comment lines. Physical lines represent the actual lines on the document regardless of what is on the lines. Measuring them consists of simply looking at the last line in the project and see what it is numbered with. Logical lines are all lines of code that contain something that is contributing to the actual programming. They must be telling the software what to do. Logical lines include code such as #include statements or other header statements that are necessary for a working piece of software. Comment lines are simply lines of code that contain comments on them. Non-comment lines are physical lines that do not have logical operators or comments on them. They usually consist of blank spaces used to organize code more neatly.[5] But as legendary software engineer Bill Gates once said, "Measuring programming progress by lines of code is like measuring aircraft building progress by weight". Maintainability generally refers to the amount of lines of code in a project, but there are



[3] https://www.altexsoft.com/blog/engineering/what-software-quality-really-is-and-the-metrics-you-can-use-to-measure-it/
[4] https://puppet.com/resources/whitepaper/2015-state-devops-report
[5] https://www.slideshare.net/dennisdegreef/measuring-maintainability-software-metrics-explained

much more nuanced ways to measure complexity and length of code. “There are several detailed test metrics used to check the complexity of code, such as cyclomatic complexity, which counts the amount of linearly independent paths through a program’s source code. The advice issued by NIST for cyclomatic complexity is that a value above 10 signifies a potentially risky codebase in terms of possible defects”[6]. The diagram above illustrates how a method like N-path or cyclomatic complexity would recognize unique program paths within a development project. The NIST method allows the reviewer to differentiate between the different between lines of code based on the flow of the code. For example, analysing how a function deals with nodes or looking at the significance of having lots of nested statements. Another more sophisticated method of measuring complexity is N-path Complexity “The NPath complexity of a method is the number of acyclic execution paths through that method”[7]. N-path complexity increases exponentially because it measures the number of unique paths a program could take so it’s easy for the numbers to get out of control. Using the N-path method requires very careful analyses and lots of planning to decide which unique paths are relevant. These different methods of maintainability all revolve around the notion that code should be easy to fix and improve. Although traditional metrics represent a large portion of the maintainability of a project, the team and project environment also contribute heavily to how efficiently products are updated.

Project metrics describe the logistics and execution of the development of software. Some common project metrics are number of software developers, the staffing pattern over the life cycle of the project, cost, schedule, and productivity. Project measures often seem straightforward as most of them only require some counting to quantify, for example, how many software engineers there are. But the only way to get use out of these metrics is look deeper into how they are related to each other. Researchers at Stockholm University released a paper about the significance of the network effect on worker productivity. The paper is particularly interesting because it delves into the how one engineer works can change how another engineer works. The paper examines certain mechanisms, such as social conformity or peer pressure, that lead to an engineer changing their work habits. The results turned out to be conclusive “We first use our exogenous network exposure matrix to study the effect of co-worker productivity on worker productivity. We show that there are indeed strong network effects in

Table 7. Network Effects from On-the-Job Training

	(4)	(5)	(6)
Local aggregate network effect, $\mathbf{H}\_trained$	0.003**		0.003**
	(0.0016)		(0.0011)
Local average network effect, $\mathbf{H}^*\_trained$	-0.010	0.006	
	(0.0131)	(0.0090)	
Worker X_i (including $trained_i$)	YES	YES	YES
Workday dummies	YES	YES	YES
Co-worker $\overline{X}_j$	YES	YES	YES
Individual fixed effects	YES	YES	YES
Team fixed effects	YES	YES	YES
Week fixed effects	YES	YES	YES
Observations	5,572	5,572	5,572
Number of person id	264	264	264

Standard errors (in parentheses) are clustered on individuals;
*** indicates significance at 1% level, ** at 5% level, * at 10% level.

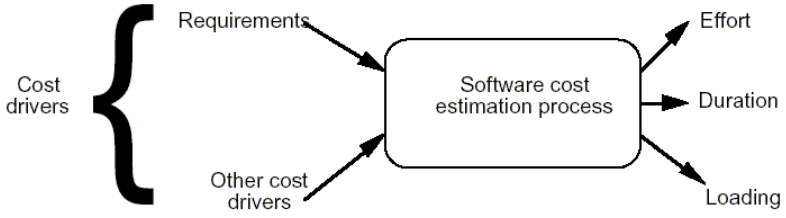
[6] Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, by Arthur Watson and Thomas McCabe.

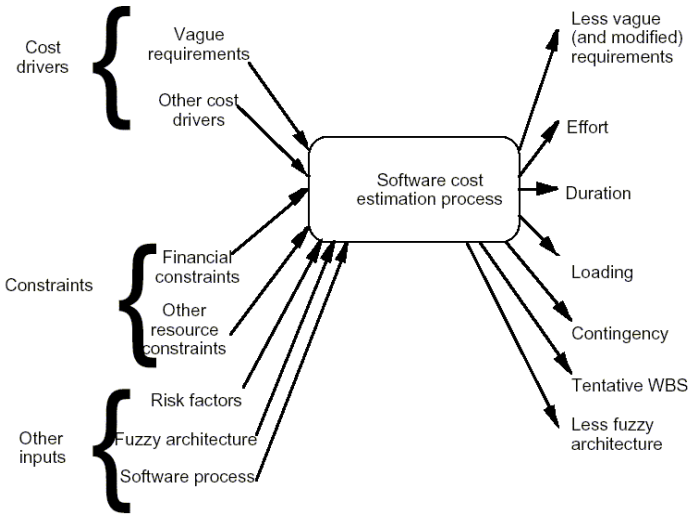
[7] https://www.slideshare.net/dennisdegreef/measuring-maintainability-software-metrics-explained

worker productivity. A 10% increase in the current productivity of a worker's co-worker network leads to a 1.7% increase in own current productivity. We attribute this productivity spill over to conformist behaviour"[8]. The network effects that the researchers describe represents the intersection of the metrics the project category. The study demonstrates that the organization of the project and the people involved directly affects how efficiently and effectively an engineer works. Choosing productive engineers and engineers that work well together directly improves the rest of the team. The network effect is also trickle down because the engineer who was made more productive now has the ability to make other engineers on the team better too. The study illustrates how some metrics can be objectively measured and documented to better a team. These findings allow teams to optimize their productivity and hire the people that will not only do good work themselves but will also inspire others to do their best work too. Another interesting finding of the study revolves around how the network itself changes workers productivity "We also find evidence of significant knowledge spill over effects from trained workers to their non-trained co-workers. Adding one additional trained co-worker to a worker's network increases that worker's own productivity by 0.7%"[9]. Adding more trained co-workers to a project can directly impact the efficiency of a new worker on the project. This finding demonstrates that the amount and type of employee staffed on a project correlates directly with the productivity of other engineers. The "spill over" effect, as the researches call it, shed light on the age-old question of training workers on the job. On the job training is especially important in software engineering because new methods and tools constantly change the landscape of the development environment. Understanding the effects of worker training can make a team significantly more productive because training needs can be pinpointed and implemented cost effectively.

Cost Metrics

Keeping track of costs can simultaneously be both the simplest and the most complex task an engineering team must deal with. The complexity begins with the process of cost estimation. There must be a benchmark to follow for a tam to judge how cost efficient a project turned out. Cost estimation normally consists of guessing how much effort a certain task or program needs for completion. The quantitative metric used for effort is the vague term person months or the amount of time for one person to





[8] Lindquist, Matthew, et al. "Network Effects on Worker Productivity." Nov. 2015.
[9] Lindquist, Matthew, et al. "Network Effects on Worker Productivity." Nov. 2015.

work for a certain period. The person months is normally weighted based on the development environment to standardize the measurement. Researchers at the University of Calgary lay out the most important aspects of estimating costs for a project "Because there are a finite number of resources for a project, all of the features of a requirements document can often not all be included in the final product… The risk of a project is reduced when the most important features are included at the beginning because the complexity of a project increases with its size, which means there is more opportunity for mistakes as development progresses. Thus, cost estimation can have a big impact on the life cycle and schedule for a project"[10]. One important aspect that is often overlooked is who does the estimating for the project. If someone outside the project, such as someone in a finance department or operations, tried to estimate the costs for the project they may forget key factors and variables that a software engineer would be aware of. The diagrams above demonstrate how the estimation process can be oversimplified if a traditional cost estimation method is used. The left diagram shows the generalized way of estimating costs that is mistakenly applied to software engineering sometimes. The right diagram illustrates the necessary process needed specifically for a software development project. The diagrams make the differences in process very obvious and they demonstrate how complicated the cost structure can get. Another facet that is mentioned in the right graph is constantly updating your cost model for things that may have come up midway through the product design or implementation. As with most aspects of the development process, there have been algorithmic approaches that attempt to automate the process. These will be discussed in the next section on platforms available to measure software engineering metrics.

Due to the explosion of the software engineering field and the constant desire by fast growing firms to optimize their operations, many platforms have been created to assist in measuring success. Firstly, there are simple social tools that can be utilized specifically within software engineering. Social tools continue to be the most popular tool used by executives to measure success "Among the technologies we have tracked year over year, social networking still

There is more potential for change without constraints.

% of respondents, n = 2,955

Entirely new process | Mostly new process

	Extent to which social technologies could enable new organizational processes, next 3 years					
	At companies with no social technology–related constraints			At respondents' companies		
Scanning external environment	33	14	47	16	14	30
Finding new ideas	32	14	46	17	13	30
Managing projects	24	15	39	12	13	25
Developing strategic plan	20	12	32	9	9	18
Matching employees to tasks	19	11	30	6	8	14
Allocating resources	18	11	29	8	6	14
Assessing employee performance	17	9	26	6	4	10
Determining compensation	12	6	18	5	4	9

[10] https://www.computing.dcu.ie/~renaat/ca421/report.html

leads the way"[11]. According to McKinsey's survey on social tools in the workplace, some of the tools companies use are online video conferencing, social networking, wikis, RSS, tagging, rating, blogs, microblogging, mashups, collaborative document editing, podcasts, prediction markets, Git, and video sharing. Executives use these social tools to keep track of conversation between engineers and gather data about how engineers work. According to the diagram to the above, Mckinsey found that respondents foresee social tools being used more to assess employee performance. While some tools are purely for productivity increases, like video conferencing, collaborative document editing, or blogs, social tools are increasingly being used to rate the performance of workers. Git is a perfect example of a platform that is social but can also be used to measure. Management can easily analyse commits, comments, and lines of code to test productivity. The social media analytics platform Klout analyses employee social media activity and gives each individual a social media a score that managers can use to judge how productive social media tools are in the workplace. Social tools are monitored by executives and used to make decisions on promotion. For example, on the messaging platform Slack, managers can set up counters related to how many times an employee adds something to the conversation or how quickly the respond to messages. Social tools are just one of the types of platforms used to monitor employee's productivity. Organizations are increasingly adopting more data-based analytics to monitor engineering output.

Platforms Available to Measure Productivity

In the past the best way of analysing engineer productivity was hiring a top tier consultant from Mckinsey or Bain and paying them to follow your employees around for a month to figure out how much work they were doing. That strategy wasn't quite working for the fast-paced software engineering industry, so companies decided to start hacking together their own DIY solutions to measure engineer productivity. These solutions worked so well other start-ups began developing larger scale "workforce" analytics software that could be sold to the entire industry. While DIY solutions and larger scale ones differ slightly, they both attempt to measure the same metrics. The most common metrics that are measure are measuring impediments, time before delivery, time in progress, time in phase, and flow control. The current top five vendors in the enterprise

HR Practice	Focus on teams? ■■■■■■■■■■
Organizational design	■■■■ Starting. Flat organizations/ holocracies.
Work Processes	■■■■■■■ Often focused on teams. Agile.
Recruitment & Selection	■ Focus on individuals. Team recruitment is scarce, even teams of two.
Workplace Design	■■■■■■ New designs take teams into account. Virtual workplace design still underdeveloped.
Talent Development	■ Most talent programs are directed to individuals.
Training & Development	■■■■ Improving team performance certainly get attention, but focus still on individuals.
Staffing/ Succession Mgt	■ Planning of individual positions. Team composition considered. Succession mgt very CEO centric.
Performance Management	■■■ Majority still on individual level, but shifting slowly to teams
Rewards	■■ Individual determined rewards still mainstream. Team/ group element in bonuses.
Internal Communication	■■■■■ E-mail still very dominant. Internal social networks slowly moving.

[11] https://www.nextlearning.nl/wp-content/uploads/sites/11/2015/02/McKinsey-on-Impact-social-technologies.pdf

workforce analytics business are SAP, Visier, ADP, Ultimate Software, and Madison Performance group. Visier, a leading workforce analytics company, put out a case study on the banking company BBVA on the benefits of workforce analytics in general and specifically Visier's technology. The case study found "Starting with workforce insights gleaned from Visier, BBVA Compass has reduced annual turnover for one key revenue producing role by 44% at its highest turnover branches"[12]. While BBVA is not a software engineer firm, the insights gathered from the case study apply to the software engineering field too. One of the most important insights gained involved the difference between HR just reporting data and HR gaining any insights into the workforce. Insights require analytics. With automated data collection and validation much more time is left for analysis of business branches and specific teams. Software engineers are often compared directly to each other in a situation when one employee has expressed interest in leaving a firm and the management is deciding whether or not to give the person a counteroffer or not. Visier's technology allows executives to compare employees side by side based on their individual attributes or compa-ratio. When looking at giving offers or cutting down labour costs analysing individual engineers remains essential, but analytics and attention has increasingly been focused on teams.

"Hierarchical organizational models aren't just being turned upside down—they're being deconstructed from the inside out. Businesses are reinventing themselves to operate as networks of teams to keep pace with the challenges of a fluid, unpredictable world"[13]. This quote, taken from one of Deloitte's Insight reports, describes a recent change in the way many organizations are structuring themselves. The above demonstrates how the recruiting and HR process has yet to catch up to the agile structure many organizations have taken on. The work process category scores a seven in being team focused while recruiting and staffing both score ones. The changes appear even more prevalently within software engineering companies because of the emphasis on agile development and structure. Essentially, companies are attempting to follow the agile development model in a business. Changing organizational structure means that human managers must alter the way the analyse employees. They must examine teams as a whole more than individual engineers.

These new requirements mean new platforms for workforce analytics. Some of the team analytics platforms available are Belbin team roles test, KeyNetiQ, Klout, OrgMapper, Part-Up, Slack, Team Effectiveness Assessment,

[12] https://www.visier.com/wp-content/uploads/2016/11/Visier-Case-Study-BBVA-Compass-Capitalizing-on-Workforce-Insights-to-Reduce-Turnover.pdf
[13] https://www2.deloitte.com/us/en/insights/focus/human-capital-trends/2016/organizational-models-network-of-teams.html

and Teamily. These platforms attempt to bring the staffing and recruiting categories closer to the work process categories. Many of the platforms stem from team role theories "According to team roles theories there are specific different team roles. These roles can be functional, organizational, personal or even skilful. Each team should consist of different team roles, depending on the specific goals the team wants to achieve. A team that does not have the ideal composition may run into problems"[14]. In software engineering the roles tend to be more specific, but the same main concept applies. For example, if you had too many idea generators in a group and not enough highly technical people, loads of ideas will be generated but they won't be turned into products that are as sophisticated as they could be. If a team has too many managers (even if they are not explicitly called managers), production will slow down and become a less productive experience. Platforms like Teamily help managers and executives pinpoint inhibitors of productivity using three main data points. Teamily uses team sentiment, work preference, and role fit to decide which engineers to allocate to certain tasks or teams. Each piece of the Teamily graph above uses large amounts of data that Teamily collects to put workers or engineers into certain categories. This type of data analytics gives a lot of power to algorithms that are responsible for making decisions about people's jobs and therefore their livelihoods. While these algorithms are undoubtedly optimizing HR tasks and assuring that people who do good work get credit for it, the technology also raises some important ethical questions about automation.

Algorithmic Decision Making

The most pressing question that we must answer as a society is "How much control do we want to give away to artificial intelligence and algorithms?". In my opinion workforce analytics in particular has the opportunity to become a tool that is abused. First, the process begins with recruitment and employers currently have an unprecedented amount of data they can use to screen candidates. For example, "In a 2017 survey of 2,300 hiring managers sponsored by CareerBuilder, 70% reported using social media to screen candidates. Furthermore, 54% reported finding information on social media that led them not to hire a prospective candidate for an open role. The most commonly cited factor was the candidate posting provocative or inappropriate content"[15]. Not that I have much to hide or be ashamed of on social media, but I treat social media as a fun space where you can joke and mess around with your friends. The more society takes it seriously the more it will turn into a ranking system for social status. If I was constantly worried about my future boss reading every single comment I write on Instagram, I would barely use Instagram at all because I have no idea what the engineering manager or boss would think of my comments. Comments and posts risk being grossly taken out of context because the analysis revolves around mostly subjective judgement. The threat of data abuse becomes more explicit after the engineer is hired for the job.

The amount of time engineers spend at their offices alone makes it extremely easy for employers to gather too much data about their engineers. One article I read mentioned that some

[14] https://www.123test.com/team-roles-test/
[15] https://www.analyticsinhr.com/blog/people-analytics-ethical-considerations/

companies take photos of their employees approximately every ten minutes![16] I think that taking constant photos of people constitutes extreme privacy infringements. I definitely don't want to be constantly documented while I'm trying to work hard on a project. I also strongly believe that this type of surveillance would lead to a toxic work environment. If engineers are constantly being monitored, they will begin to distrust management and the agile team development strategy would break down. I also think it could create an uber competitive culture between engineers. Once engineers figure out what types of things the management is looking for, they will constantly try to beat their fellow engineers in those categories. I also think that allowing algorithms to decide some outcomes will put an unbalanced focus on certain attributes. Putting unnecessary emphasis on certain qualities will incentivize all workers to abandon their own habits and attempt to conform to the algorithm's idea of productivity. Although it is useful to gain insights on how teams function, it is also paramount that workforce analytics do not take the individuality out of each engineer. One engineer might have a slightly different style of work that does not fit in any boxes for metrics of productivity or another engineer might not comment as frequently because they like to make less frequent, dense comments. The types of projects each team is working on could also taint the productivity metrics. Every project is different and requires different types of engineering strategies to complete efficiently. If the project does not meet the algorithmic standards set by the data and management, how will management take into consideration that the teams could have been assembled more efficiently. The management that sets up the algorithm needs to program in their own mistakes to make it fair. Simply based on human nature, the management will generally grade themselves less harshly and be tempted to defer blame to the engineers explicitly working on the project. I strongly believe all these considerations, and more must be scrutinized strongly before algorithmic approaches take more responsibility in making HR decisions. In the meantime, the General Data Protection Regulation (GDPR) has set forth some basic guidelines regarding all types of privacy.

Where does GDPR Fit in

GDPR is a privacy and data protection regulation that attempts to protect consumers from losing control of their data. It heavily focuses on transparency and not allowing companies to collect data without the consent of consumers. While lots of it focuses on privacy for consumers being targeted by large scale algorithmic advertising targeting, there are many parts that pertain to workforce analytics ethics. Unlike previous data regulations, GDPR explicitly confronts the issue of algorithms making decisions "The data subject has the right not to be subject to a decision, which may include a measure, evaluating personal aspects relating to him or her which is based solely on automated processing"[17]. In the previous paragraph, I discussed my personal opinion about engineers' rights to choose if an algorithm judges them. I think this type of regulation is important in setting boundaries for these types of data collection. The productivity measures that I reported on earlier in the report are certainly important in improving the engineering process and creating better products, but that shouldn't mean peoples privacy needs

[16] https://www.analyticsinhr.com/blog/people-analytics-ethical-considerations/
[17] https://phoebevmoore.wordpress.com/2017/11/07/the-gdpr-algorithms-and-people-analytics/

to be infringed upon. Another large part of GDPR revolves around getting consent from people about taking their data. While consent can be relatively straightforward when asking a webpage visitor if their data can be collected, getting consent in a development environment can be difficult because engineers may feel pressure to succumb to the practices of management. GDPR realizes this and explicitly addresses it "public authorities and employers will find using consent difficult… employers and other organisations in a position of power are likely to find it more difficult to get valid consent"[18]. Consent becomes very difficult in the workplace because it is impossible for engineers not to think they will be judged differently if they don't comply to the data collection practices. GDPR does a good job at starting the conversation about regulation related to workforce analytics, but there are still some significant loopholes that could cause problems for engineers. For example, employers may process data without consent if "processing is necessary for the performance of a contract with the data subject or to take steps to enter into a contract, for the purposes of legitimate interests pursued by the controller or a third party, except where such interests are overridden by the interests, rights or freedoms of the data subject"[19]. These clauses in the regulation could be used by management to analyse engineers based on things they are not aware of. Young engineers, such as I, will increasingly be asked to develop software to assess and analyse not only other software engineers, but other members of the labour force. GDPR started the conversation about how we will treat this new era of algorithmic decision making, but we are far from a solution.

Conclusion

Over the past 20 year the software engineering industry has quickly become one of the most important and most scrutinized professions in the modern economy. The pressure to constantly improve and profit has led to an obsession with measuring and optimizing software engineering performance. The study of productivity and the software engineering process will allow the industry to improve to the ever-evolving standards of technology. The creation of platforms to measure productivity has quickly become a new and essential part of the software industry. These algorithmic solutions will undoubtedly help improve performance and guide the software engineering world to new heights. However, we, as software engineers, must stay vigilante to the threat of decisions becoming completely autonomous or left to a small group of executives outside the circle of engineers. As technology becomes more and more intelligent, we as a society must constantly be aware of the ethical challenges that will surface constantly. In this age of intelligent machines, there has never been a more important time to trust our gut feelings.

[18] https://phoebevmoore.wordpress.com/2017/11/07/the-gdpr-algorithms-and-people-analytics/
[19] https://phoebevmoore.wordpress.com/2017/11/07/the-gdpr-algorithms-and-people-analytics/