Kaitlyn Walsh

Professor Barret

Software Engineering

20 November 2018

Measurement and Ethical Methods of Software Engineering

In the world of Software Engineering, there is controversy on how you should measure software engineering, as well as controversy over if it should be measured at all. Unlike other professions, there are few who are capable of actually measuring something as complex as software engineering and few ways to accurately measure software engineering. But despite the complications, measuring software engineering is important for keeping all software engineers ethical in their decision making and for producing better software to increase business productivity.

Creating a code of ethics which encapsulates all of software engineering is not an easy task. There are still scholars who do think it cannot be accomplished. The Institute of Electrical and Electronic Engineers (IEEE) Computer Society created eight primary principles that create the "Software Engineering Code of Ethics and Professional Practice", as a way of holding all professional software engineers to the same standards. Their hope was to create a code that allows everyone to make the same ethical decisions when creating new products, and to provide guidelines for answers when software engineers are faced with ethical decisions. The principles focus on eight primary categories:

1. Public- All code produced by software engineers shall act consistently with public interest
2. Client and Employer- Software Engineers shall act in a manner that is in the best interests of their clients/employers, only if it is consistent with public interest

3. Product – Software Engineers shall hold their products and modifications to meet the highest professional standards possible at the time
4. Judgment – Integrity and independence shall be maintained in professional judgment
5. Management- All software engineers shall follow and promote an ethical approach to the management of software development and maintenance
6. Profession – Software Engineers shall advance the integrity and reputation of their profession consistent with public interest
7. Colleagues – All colleagues shall be fair and supportive of each other
8. Self– Software Engineers shall participate in lifelong learning while practicing their profession and promote a continuing ethical approach to their practice

While the guidelines given by the IEEE Computer Society are nice in theory, they do not encompass every possible situation. Engineers still must use their best judgment to decide if a choice is the most ethical way to create a product. The primary focus of the ethical guidelines is to keep the public in mind. At the end of the day, software engineers work to benefit the people, not only their clients. All software created affects everyone, not just clients or employers. Because software development can become very complication, many believe the only way a software engineering process is truly ethical is when the client or stakeholder fully understands every possible challenge and failure of the program. If a client or stakeholder agrees to a program without fully understand possibly implications, they are at risk of not only losing their money, but of being held liable for risks they were unaware of. This perspective is interesting, because clients will rarely understand the full extent of what the program does. If they did, it is very likely that a client would have produced the program themselves. But realization of how a decision can be fully ethical should act as encouragement to engineers to not over complicate systems, and explain the work to the best of their abilities before handing over a finished

product. If a client cannot understand a majority of their new product, then perhaps the product is too complicated and could theoretically be simplified to work more efficiently.

Technology is used everywhere today. From laptops to smart watches, and now even smart cars, there is no way to avoid it. When a car company chooses to use a certain program for their brakes, navigation, security features or even music, the company is trusting the creators of the program to produce the most reliable piece of software they are capable. When the car has an accident because of a glitch in the brake, navigation, or security features, the car company is held responsible. The company is at fault the software glitch, not the creator of the product. The software engineer behind the product is rarely, if ever, named in the papers, as the reason behind the accident, despite the fact it was their direct work which could have caused it. Engineers are often disconnected from effects of their work, which is a reason for ethical concern. When tragedies occur as a result of a malfunction in software, they need to be aware so it doesn't happen again.

Ethics need to have a play in software engineering because technology has taken of daily life. It is impossible to go a full day without interacting with a piece of technology. While customers may never fully understand the entire process of how and why a piece of technology functions the way it does, Software Engineers must help users understand it to the best of their abilities. The more you know about something you use, the easier it is to use it. And when technology fails a user, it is easier to fix when you understand even just the basics of the system. The ethics "code" of software engineering will never be perfected, and I believe the controversy will continue. But I think a majority of software engineers truly use their skills to create products that benefit society and with the intent to do good, not harm.

The current ethical situation in the software engineering world is interesting. In the past few years, I believe an agreed upon universal ethical code would have been useful. In the United States, there was a complication between the FBI and Apple. The FBI had acquired the phone of a suspect involved in a shooting. They believed there was vital information on the iPhone, which could inform them of the criminal's motive and potential future attacks. When asked to unlock the iPhone, Apple refused. They claimed that unlocking the phone would be a breach in customer security. Apple thought that by creating a universal iPhone password breaker for the FBI, they would be compromising the security for all users. Therefore, Apple refused to help the FBI. Eventually, the FBI unlocked the suspect's phone without Apple's help, but this raised many concerns. If there is a way to create a universal password for iPhones, how long will it be until all passwords become irrelevant? Until a hacker discovers this universal passcode. A universal passcode shows there is a flaw in Apple programming. Should Apple have made it their priority to discover and fix this flaw? Is it the FBI's ethical duty to inform Apple of this flaw? Some would say it is, but others would say the FBI has a greater ethical responsibility to gain this information and make a case against the suspect. The matter could come down to the "greater good" defense. Are more people protected now that the FBI has this information, or will more people be protected if Apple resolves the flaw? Situations such as this one do not bend to the simplicity of the IEEE's ethical framework. They are simply too complex. Ethics will never be black and white. Situations challenge everything that has been put into place. Human nature is not rational and should not be treated as such. While having a code of ethics and core principles to follow are a good start to adding ethics in software engineering, they simply cannot cover every situation that could occur in the world. The increasing use of technology ensures this. But a code of ethics and universal promise to adhere to it as closely as possible could bring software

engineering to a position where situations become more similar and easier to solve. The code is not perfect, but neither is the process of software engineering.

While software engineers should work with a specific code of ethics, one must be able to measure the work in order to determine if it is being created for good or not. Standard methods of measurement do not work efficiently in software engineering. For example, some companies measure the quantity of work produced in a single week. This does not translate well for software engineering, where the best way of measuring quantity of work would be to measure lines of code produced per week. But the best code is the code that is not long in length. The most concise, uncomplicated code runs the best and takes up the least amount of memory. For this reason, direct measurements do not work. Lines of code, churn, and meeting deadlines are all things a software engineer have the ability of producing and meeting. But just because they can, does not mean they should. Often their quality of work declines because of these measurements.

Therefore, measuring software engineering must be done by measuring other variables. Some suggested metrics to measure have been success, customer satisfaction, happiness and impact. The success of a program can be measure by different factors—how well a program runs, the amount of times the program crashes, the time it takes to fix bugs in the program. Measuring the success of a product allows the company to see how well software engineers are performing. If a product is successful, with minimal bugs or bugs with easy fixes, then the software engineers are doing their job properly. This form of measurement also shows constant improvement on products, as new bugs arise over time with software updates, and shows a consistent dedication to the improvement of engineering as stated in discussion of ethics.

Customer satisfaction should also be taken into consideration, either by itself or as part of success, when deciding measurements. Software Engineers create products for customers, therefore a customer's response should be an excellent judge of measurement. When looking for a product, customers know what they want. They set out on a mission hoping to find something that meets all their criteria. Sometimes they get exactly what they want; sometimes they do not. But that could be for multiple reasons. Customer reviews give a description of their experience with the product. Their overall experience at a glance seems harsh—just a number of stars. Their explanations must be used to create a better understanding of the product. Some customers are not satisfied with products because they do not do what the customer expected them to do. In this scenario, the customer's dissatisfaction can be shared between both the company and the customer. While the product may not have been advertised well, customers should do more research on products before purchasing them. Customer feedback tells if the product was faulty or simply not the right fit for them. It helps the company figure out what products need fixing, how to make their products easier to understand, and what people expect out of their product. These comments help the company make newer and better versions of their products; they help the company improve their customer base. Customer satisfaction is easy to measure. People are always willing to give their opinions. But the science is not perfect, like anything else. People have bias and many have opinions backed up by nothing.

Besides measuring the success of a business or customer satisfaction, it is possible to measure the happiness of employees in a company. A happier employee, in all fields—not just software engineering— is a better employee. A study shows that happy employees are 37% more productive than those who are unhappy and have a 300% increase in creativity. More productivity and creativity creates a better software engineer. They are able to find better

solutions to problems occurring in programs and create programs that solve more issues more efficiently. A company with happier employees has a better reputation and brand. These companies are known for taking care of their employees, have an increase in the buying of shares, and increase profit on all fronts. Not only would it be a business savvy move to increase employee happiness, but everyone in the company would benefit. Theoretically, a company which wants to improve happiness, would improve working conditions in company offices. This includes better office temperatures, break rooms, office spaces and office atmosphere. This could increase creativity in software engineers, which would improve the field in general. Constant improvement in engineering would keep industries moving forward and would help industries reach their maximum potential.

A final sufficient way of measuring software engineering would be to measure the impact of the code. To do this, software engineers, or the companies hiring these engineers, would have to look at the effects of changes to the code. How does changing one line alter the program? Does changing three lines of code make more sense than altering the files that interact with the code? These are questions that would need to be asked before measuring the impact of code. If changing a few lines of code makes the software more universal, then it makes sense to change them. The program will be more useful to clients and to the software engineer. Measuring impact is also a good way of determining how efficient the code is. If changing a line makes almost no difference in the code, then is that line truly necessary? Could the code be condensed into something more efficient and condensed? Maybe, the team needs to take some time to churn the program before adding more. Either way, measuring impact would help managers see what is going well and what needs improvement.

As a disclaimer, one set of metrics cannot work for all companies and software engineers. Each company should figure out their primary focus and chosen goals. Then they should figure out which measurement style matches those goals. When employees know a particular thing is being measured, they often change their work ethic to increase that metric. Because of this change though, metrics must be trusted to a certain extent. Measuring lines of code or deadlines, for example, could make code messy. Any software engineer can meet a deadline, but their work suffers. The related changes cause by measuring success or happiness levels do not seem to affect overall results. There is no way to measure success without success actually being present. There is no way to fake happiness levels. Stress, anger, and anxiety are hard to hide and hard to fake. Therefore, these measurements appear to be the most accurate measurements to handle, despite not measuring software engineering directly.

With so many areas for measuring software engineering, some may still ask why it is necessary and if measuring it is even worth the effort. To those people, I say look at all the businesses that use technology in their day to day life. Not only do businesses use technology for managing operations, but they also use it for marketing purposes as well. Today, at least 38% of businesses use social technologies to interact with their customers. Social media platforms like Facebook, Instagram and Twitter make it easy for their users to control to see advertisements and links to businesses. Social technologies also allow small businesses to grow. Sites like Etsy give small business owners a platform to make their products accessible worldwide, something which did not seem feasible without a big corporation a few years ago. Businesses who use technology usually see a 20% increase in revenue. They are more successful and probably more efficient. Things like digital “clouds” allow companies to have back-ups of their data in case something ever happens to their physical machines, cutting down recovery time after physical attacks.

While there are many things to be measured, some companies are using more than just technology to measure success and business improvement. These companies are embracing technology on a completely different level; they are actually incorporating game techniques, or game theories, to create a work environment that motivates employees as if they were in a video game. In an article by the Wall Street Journal, it explains how "game theory" was used in companies to promote better social habits and create a healthy competition in the office. The theory acts similarly to "missions" in video games, where the hero is sent on side missions to help the local man gather all of his chickens or items, to gain incentive points before moving on to the primary objective of the level. But instead of gathering chickens, employees are encouraged to carpool to work or boost their sales goals and increase their quota, which is then displayed on a board or screen for the entire office to see. This form of measurement uses software engineering as well as forms of direct measurement. The use of game theory in the office affects more than just numbers. The success of the company improves with the happiness of employees. People love a healthy competition and having fun at work. The Wall Street Journal article said an employee even made invoicing fun, which is something that has never been associated with invoicing. An increase in fun leads to an increase in happiness levels. Therefore, while many researchers are looking at how to measure software engineering, few are looking at how to use software engineering to measure engineering. Though slightly unconventional, game theory seems to have a significant effect on the workplace, without using other expensive technology, like badges that measure physical activity or track employees' locations in the workplace. While items like these badges are useful for companies, they seem more useful for research purposes instead of actual improvements. I believe game theory

solutions should be placed in effect after measurements have been taken, in order to have the theory produce ultimate results. For game theory to work, concrete goals need to be determined.

Game theory is just one way to use the data accumulated by measuring engineering. There are other ways to analyze the data, but the primary goal is to use the data to create new ways of improving companies. The data should show flaws in the company and systems, but by creating new goals the company will be able to create new solutions to solve issues. While game theory may work for some companies, it won't work for others. New solutions have to be put in place, such as changing the office environment, promoting moving every now and then, and other factors that could improve numbers or happiness levels.

Therefore, we have determined the ethics of measuring engineering, ways to measure engineering and what do to with the data accumulated by measuring engineering. The debate on whether software engineering should be measured will probably continue. There is not a single answer that perfectly solves the controversy. There is also a lack of people in the world who could directly measure software engineering and produce trusted results, which is a flaw for the argument that engineering should be measured. The number of software engineers, who are capable of this task, is not great enough. Therefore, measuring software engineering indirectly is the only way at this time. Through indirect measurements, such as success and happiness, you can really measure the effects of software engineering, more than software engineering itself, which might be a better way of judging software engineers.

References

"9 Metrics That Can Make a Difference to Today's Software Development Teams", *TechBeacon*, accessed 07 November 2018, <https://techbeacon.com/9-metrics-can-make-difference-todays-software-development-teams>

Akhnoukh, Nader. "You CAN (And Should) Measure Software Engineering Performance", *Kapost Engineering*, 25 August 2015, accessed 11 November 2018, <http://engineering.kapost.com/2015/08/you-can-and-should-measure-software-engineering-performance/>

Benner, Katie. Markoff, John. Perlroth, Nicole. "Apple's New Challenge: Learning How the US Cracked Its iPhone", *The New York Times,* 29 March 2016, accessed 02 November 2018, <https://www.nytimes.com/2016/03/30/technology/apples-new-challenge-learning-how-the-us-cracked-its-iphone.html>

Bughin, Jacques. Chui, Michael. "Evolution of the networked Enterprise: McKinsey Global Survey Results", *Next Learning,* accessed 11 November 2018, <http://www.nextlearning.nl/wp-content/uploads/sites/11/2015/02/McKinsey-on-Impact-social-technologies.pdf>

Duffy, Rob. "Software Development: A Better Way to Measure Success", *DevOps.com,* 15 June 2018, accessed 05 November 2018, <https://devops.com/software-development-better-way-measure-success/>

Editorial Team. "For Engineering Performance, Stop Measuring Productivity", *WorkSpace Today,* 13 January 2017, accessed 11 November 2018, <http://theworkspacetoday.com/2017/01/13/for-engineering-performance-stop-measuring-productivity/>

Kamthan, Pankaj. "Ethics in Software Engineering", *Concordia University,* accessed 11 November 2018, <https://pdfs.semanticscholar.org/caa6/07b7a74db0a71fd66b21fe0c8643fad7e3f7.pdf>

Kaner, Cem. Bond, Walter. "Software Engineering Metrics: What Do They Measure and How Do We Know?", *International Software Metrics Symposium*, 2004, accessed 10 November 2018, <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=96403470F6E0EF15C14F48EE2566B3A5?doi=10.1.1.1.2542&rep=rep1&type=pdf>

Kimmel, Travis. "5 Developer Metrics Every Software Manager Should Care About", *Git Prime,* accessed 08 November 2018, <https://blog.gitprime.com/5-developer-metrics-every-software-manager-should-care-about/>

Lurie, Yotam. Mark, Shlomo. "Professional Ethics of Software Engineers: An Ethical Framework", *Science and Engineering Ethics*, June 2015, accessed 11 November 2018. PDF.

Oliveira, Edson. Viana, Davi. Cristo, Marco. Conte, Tayana. "How have Software Engineering Researchers been Measuring Software Productivity?", accessed 09 November 2018, PDF, <http://www.scitepress.org/Papers/2017/63144/63144.pdf>

Silverman, Rachel Emma. "Latest Game Theory: Mixing Work and Play", *The Wall Street Journal,* 10 October 2011, accessed 03 November 2018, <https://www.wsj.com/articles/SB10001424052970204294504576615371783795248>

"Software Engineering Code of Ethics". *IEEE Computer Society*, 2018, accessed 12 November 2018 < https://www.computer.org/web/education/code-of-ethics>.

Yano, Kazuo. Akitomi, Tomoaki. Watanabe, Junichiro. Tsuji, Satomi. Sato, Nobuo. Hayakawa, Miki. Moriwaki, Norihiko. "Measuring Happiness Using Wearable Technology", *Hitachi Review,* 2015, accessed 11 November 2018, <http://www.hitachi.com/rev/pdf/2015/r2015_08_116.pdf>