MEASURING SOFTWARE ENGINEERING

PRAPTI SETTY | STUDENT NUMBER - 16336726 | CS3012 – SOFTWARE ENGINEERING

TABLE OF CONTENTS

INTRODUCTION 2

WHAT IS SOFTWARE ENGINEERING?

WHO IS A SOFTWARE ENGINEER?

WHAT IS MEASURING SOFTWARE ENGINEERING?

MEASURING SOFTWARE ENGINEERING 3

MEASURABLE DATA

COMPUTATIONAL PLATFORMS TO PERFORM THIS WORK

ALGORITHMIC APPROACHES

ETHICAL IMPLICATIONS OF ANALYSING ENGINEERING 9

BEST TECHNIQUE TO MEASURE SOFTWARE ENGINEERING 10

REFERENCES 11

INTRODUCTION

WHAT IS SOFTWARE ENGINEERING?

Software Engineering is the term that is believed to have first been coined by Margret Heafield Hamilton. The idea of naming this discipline stemmed from aiming to giving the field legitimacy. Her purpose was to earn the discipline acceptance as being an engineering field in its own way.

There are many recognized definitions of Software Engineering. For the purpose of this report, I will define software engineering as 'The process used for development of software in a systematic, disciplined method based on engineering principles and recommended practices.' It is important to note that there is a difference between programming and software engineering. Technopedia compares programming and software engineering by stating the following in their article: "In contrast to simple programming, software engineering is used for larger and more complex software systems, which are used as critical systems for businesses and organisations." [1]

WHO IS A SOFTWARE ENGINEER?

"A software engineer is a person who applies the principles of software engineering to the design, development, maintenance, testing, and evaluation of computer software.

Prior to the mid-1970s, software practitioners called themselves computer programmers or software developers, regardless of their actual jobs." [2]

A software engineer relies on the practises and principles of software engineering to write software or change existing software.

WHAT IS MEASURING SOFTWARE ENGINEERING?

In this report, I will discuss the ways in which software engineering processes can be measured and assessed in terms of measurable data, an overview of computational platforms available to perform this work, the algorithmic approaches available, and the ethical concerns surrounding this kind of analytics. I will be examining some of the texts on the reading list [3] available for the module along with other research papers and articles available online to determine the measurement of software engineering.

MEASURING SOFTWARE ENGINEERING

Using software metrics has had its successes and its failures. It has been a remarkable accomplishment if we judge it by the books, articles, research projects. It is fascinating to note that many companies, small and large, have begun including software metric programmes.

MEASURABLE DATA

For years, academics have looked for analytics that help recognize and comprehend development procedures and measure the rate of productivity on engineering projects and in engineers. Researchers aim to produce numbers and statistics that characterize properties of software code and software quality. “Unfortunately, the easier an analytic is to collect and the less controversial it is to use, the more limited its usefulness and generality...” [4]

Easily obtained analytics can be useful as rough guides and may be used for estimations or approximations but may often have little to no impact when requesting accurate data to improve processes as outlined in the article from which the quote above was taken.

“It is fair to say that the most significant [metric] to provide [is] information to support managerial decision-making during software development and testing.” [5] According to this statement, we can place an emphasis on and determine that the most noteworthy metrics are ones that have an impact in management. This article resounds some of the same arguments as the previous one stating that “the metrics being practised are essentially the same basic metrics that were around in the late 1960s. In other words, these are metrics based around LOC (or very similar) size counts, defect counts and crude effort figures (in person months).” It adds on the argument that “simple metrics … are easy to understand” and clarify that in the article they will “use primarily these simple metrics to build management decision support tools that combine different aspects of software development and testing and enable managers to make many kinds of predictions, assessments and trade-offs during the software lifecycle” but they explain “traditional (largely regression-based) models are inadequate for this purpose.” These are interesting statements in an intriguing article that uses simpler approaches to build more interesting metrics through data collection.

So what data can we collect, and can we really measure software engineering through data collection accurately by analysing this data?

Lines of Code (LOC) is a metric that is very easy to collect, but it is debatable whether LOC measurements are useful in languages other than line-oriented languages. Comparisons involving only the order of magnitude of lines of code in a project are generally more useful. For example, comparing a 10,000-line project to a 100,000-line project is far more useful than comparing a 20,000-line project with a 21,000-line project. To add to this, LOC is particularly ineffective at comparing programs written in different languages unless adjustment factors are applied to normalize languages. The issues don't stop here, a developer's experience may lead to the number of LOC they write. In the industry, the 2 main types of LOC measurements are Physical LOC (PLOC) and Logical LOC (LLOC). A developer with more experience may be able to summarize a few LLOC into one PLOC whereas a developer with a lack of experience may require so much as a few PLOC for a single LLOC. In the long run, while LOC may be an easy way to express size of software and visualize productivity through automated counting bots and scripts, it is not an accurate method in measuring productivity and engineering.

A metric that I found fascinating when researching collectable data to measure engineering that is easy to collect measurements for the Uptime [6] of a product. In an age where almost everything we do requires us to use the internet, many companies are focussed on building products to be used on the internet, and measuring Uptime is not difficult. What is Uptime? For a product on the internet, how much of the time is it available to customers as opposed to not available. This measurement is often a percentage. Why is Uptime a good measurement? Every time your product may go down, it is a distraction to the engineering team. As a measurement, Uptime allows engineers to determine the time a system can be left unattended without crashing or need for maintenance. If the uptime is very low, the engineering team may be focussed on bringing the product back up and this will be an extra cost to your business. There are many methods to measure Uptime and these depend on the type of product, but a simple calculation used as a baseline in the industry is measuring the availability of a product in hours per day (can be scaled based on robustness of product).

COMPUTATIONAL PLATFORMS TO MEASURE ENGINEERING

Along with all the time and money invested to research the topic of measuring software engineering, a number of platforms were designed. The platforms do not just take single measurements and attempt to predict engineering performance and productivity but rather they take a series of figures and use these to forecast, often with amazing accuracy, the performance and efficiency. Different platforms will produce results in different fields by applying different methods of measurements.

Many companies and businesses choose to create their own platforms in the form of dashboards to measure their engineering and the success of projects and products by choosing some KPIs. What are KPIs? KPI stands for Key Performance Indicator. KPIs are a set of measurements meant to indicate whether a product is successful or not. The advantages of creating your own platform based on several previously tested and used measurements are numerous. As each project is unique, the KPIs they choose may be the ones that when combined are tailored to the project and will be able to help them track their engineering and expand their businesses. Some of the more generic KPIs applicable to most software engineering businesses/products are Process Metrics, Quality Metric and Technical Metrics [7]. Calculating figures to represent Response Time, Product development cost, Customer Satisfaction, Engineering changes, Maintenance required and so on will help every software project form a platform for measuring engineering and success. These serve as significant pointers for businesses should they choose to base their platform of measurement of engineering on KPIs.

There are some possible disadvantages when using KPI dashboards as a method of measuring software engineering. Organizations often end up adopting industry recognized KPIs blindly not realizing that KPIs are only as useful as a business makes them. Without tailoring the KPI dashboard to a projects specific needs, the metrics prove almost impractical. Thus, KPIs have begun to gain a negative connotation associated with them and have fallen out of favour for some.

Capability Maturity Model Integration (CMMI) is a process level improvement training and appraisal program [9] often referred to as a model for optimizing development processes. It helps organizations streamline procedure development, promoting a creative, effective

culture that decreases risks in software, product and service development. CMMI originated in software engineering but has been generalized over time to include other areas of interest, such as the development of hardware products, the delivery of all services, and the acquisition of products and services. The CMMI was developed to combine multiple business maturity models into one framework. For businesses that choose CMMI method od measuring engineering, the goal is to bring the business up to the final level, 5, the "optimizing" maturity level. The five maturity levels are Initial, Managed, Defined, Quantitatively managed and Optimization. Businesses that reach Level 4 and 5 will begin to focus their time on maintenance and regular improvements.

The Personal Software Process (PSP) is a software development process that was designed after the CMMI process began to be used in the industry and is more focussed on software engineering. While the CMMI process is still used rigorously in the defence departments, for software products a tool like PSP proves more helpful.

PSP is a process structure that intends to help software engineers better understand and improve their performance by tracking their predicted and actual development of code [10] with a disciplined personal framework. The PSP helps software engineers to improve their estimating and planning skills, make commitments they can keep, manage the quality of their projects, reduce the number of defects in their work. In other words, it shows them how to plan their work, use a clear and measured process, create assessable goals, and track performance against these areas. The PSP shows engineers how to accomplish excellence from the beginning of the job, how to analyse the results of each job, and how to use the results to improve the process for the next project.

The PSP platform of measurement has four core measures in its system:

- Size – the size measure for a product part, such as lines of code (LOC).
- Effort – the time needed to complete a task, usually noted in minutes.
- Quality – the number of defects in the product.
- Schedule – a measure of progression, tracked against planned and real completion dates.

Applying standards to the process can ensure the data is accurate and reliable. This data can be logged in forms, normally using a PSP software tool.

ALGORITHMIC APPROACHES

There are also some quick algorithmic approaches frequently used to measure software engineering. Using these algorithms mean that businesses can track their performance for smaller sections of time than a year and make changes quickly as needed.

What is an algorithm? In simplistic terms, a set of rules used to solve a problem. In this instance, by collecting some data and applying some calculations to it in the form of an algorithm, a team can measure its success with the intent of cultivating efficiency and driving performance [8].

There exist simple algorithms to measure research and development (R&D) engineering, engineering sustainability and engineering cost. By choosing the algorithms that help the team working on a project or in a business analyse their engineering, the team can create their own charts and graphs and use these as indicators for projects and determine when management changes need to take place.

R&D is a cost that can often end up expensive for a company and therefore having an algorithm in place to measure the success if the money invested in R&D often proves to be a good idea. A company may spend a huge sum on R&D and then choose not to innovate the idea, or a competitor may release a similar product sooner. The simplest algorithms to measure these are:

$$\frac{\text{R\&D (NPI) cost for the year}}{\text{Value of the Product Being Developed}} \qquad \frac{\text{R\&D (NPI) cost for the year}}{\text{Price that the Equipment Developed is Expected to Bring}}$$

R&D cost for a year may include items like salaries, testing costs, consulting costs and support costs. By analysing these measurements year on year, companies can determine if the money being invested into R&D relative to value of the product and the profit it may bring is worth it for them and whether they think they should continue investing.

Measuring the cost of supporting existing products is also very important when it comes to a software project. In certain instances, maintain products may prove non-lucrative to projects as with ever changing technology and software, more bugs may become apparent and unavoidable. Teams may focus on creating solutions to the bugs and issues and there

may be losses generated instead of profits when trying to maintain an old product as revenues may decrease over time with competitors releasing similar products. If many bugs and problems occur, teams may not be able to focus on improving the product and adding new features, driving customers to move to competitors. There are several algorithms that can be used to measure cost of supporting existing products that a company may use. They can also choose to come with their own algorithm specific to the needs of their product and engineers by referring to and tailoring the algorithms outlined above.

ETHICAL IMPLICATIONS

Along with any form of analysing productivity, come the ethical implications associated with analytics. What is an ethical implication? Ethics refer to a subjective philosophy dealing with right and wrong or good and evil. Ethical implications are the result be viewed with those ethical considerations in mind, but forecasting ethical implications is difficult because human psychology and culture is so complex and nuanced. In general trends may be predicted, but we cannot fully understand the possible implications of implementing processes to measure engineering.

Say for example, a company was to begin measuring engineering productivity through Lines of Code. Engineers may falsely believe that they could get a raise in pay based on the number of LOC. This may motivate them to be less efficient with space managements and rather careless with their code so as to improve their LOC count. In this process, more experienced engineers may end up losing out as they may be able to merge a few lines of physical code into a single line and their space management may not be rewarded fairly.

When it comes to discussing the ethics of using a system like CMMI or PSP, or algorithms to measure engineering, there are less implications with regards to singular persons and more to an overall organisation. While business should set goals and targets and measure engineering. They should not do this at the cost of their own employees' health and happiness. Driven solely by numbers and statistics, and a focus on highest achievable profits, a business may be regarded as unethical to work for or with as they may be motivated to find a means to justify the end goal like sell private data, overwork employees, etc. Without a consideration for their employees, they will likely lose their reputation in today's world where a growing emphasis is placed on the safety of, and the regulation around employee benefits.

BEST TECHNIQUE TO MEASURE SOFTWARE ENGINEERING

Based on all the readings and the research I have done to write this report, accounting for the ethical implications of analysing data, it is very easy to determine that measuring engineering is no easy task, but it can be done. There is no singular method or model that is guaranteed to work for every software project or company, but a combination of methods may allow for a company to help itself grow and expand at a rate that is optimum for it.

Creating their own personalized dashboards based on previously recognized KPIs combined with algorithms and simply measured data to track research and development and engineering in businesses and projects is a method that is being adopted quickly within the industry. The advantages include the uniqueness of such a system allowing for it to be tailored precisely to a company's needs.

REFERENCES

[1] https://www.techopedia.com/definition/13296/software-engineering

[2] https://en.wikipedia.org/wiki/Software_engineer

[3] https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/reading_workmeasurement.html

[4] Searching under the Streetlight for Useful Software Analytics - Philip M. Johnson

[5] Software metrics: successes, failures and new direction - Norman E. Fenton, Martin Neil

[6] https://stackify.com/measuring-software-development-productivity/

[7] http://ryeok.com/blog/2014/2/1/5-key-metrics-for-engineering-departments

[8] https://www.apriori.com/blog/6-metrics-that-will-help-improve-your-engineering-productivity/

[9] https://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration

[10] https://en.wikipedia.org/wiki/Personal_software_process