Measuring Software Engineering

CSU33012

Software Engineering

Stanislus Igboeli - 18341643

CONTENTS

INTRODUCTION

The aim of this report is to:

- **Clarify as astute as possible what exactly software engineering is and all it entails.**
- **Possible ways to measure it.**
- **Probable interpretations of the resulting data.**
- **And finally to answer whether or not we should even though we can.**

What is Software Engineering?

Since its inception Software Engineering has been defined as a process of analyzing user requirements and then designing, building, and testing software application which will satisfy those requirements.

Let's look at the various definitions of software engineering:

IEEE, in its standard 610.12-1990, defines software engineering as the application of a systematic, disciplined, which is a computable approach for the development, operation, and maintenance of software.

Fritz Bauer defined it as 'the establishment and used standard engineering principles. It helps you to obtain, economically, software which is reliable and works efficiently on the real machines'.

Boehm defines software engineering, which involves, 'the practical application of scientific knowledge to the creative design and building of

computer programs. It also includes associated documentation needed for developing, operating, and maintaining them.'

Due to the complexity of the term and what it refers to as well as all it encompasses, the exact definition is one that varies based on who and where you ask. For the purpose of this report, the aforementioned characterizations should suffice.

Why We Gather and Analyze Data.

There are many reasons as to why we gather and analyze data in software engineering. In this section of the report I'll focus primarily on the accumulation of information by employers and the measures they hope to take with garnered Intel to create the most optimal work environment possible for their businesses.

Not dissimilar to the lack of a standard definition for Software Engineering is the lack of a standard definition for Software metrics which are valued by all software development teams. As a result of the variability of Software Engineering it follows logically that different attributes are valued at different degrees depending on the concerned Development team. Therefore, this report will focus majorly on three foundational types of metrics which can be used to assess any software project and development team.

- **i.) Process Metrics**
- **ii.) Project Metrics**
- **iii.) Product Metrics**

Product metrics describe the characteristics and attributes of the desired finished product such as its size, complexity, performance, overall functionality and quality level.

Process metrics refers to measures taken which can be used to improve the effectiveness and efficiency of the software's development and maintenance. These include: Time taken to identify defects, Actions taken to minimize the quantity of defects or actions taken in response to a defect and the time taken to correct or fix defects

Project metrics describe the project traits and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

It is important to note that some metrics belong to multiple categories. For Instance, the in-process quality metrics of a project are both process metrics and project metrics.

In essence it is vital that we measure software as the resulting data can enable the head of a development team to not only identify possible issues but also minimize potential risks if not eliminate future failures associated with their project(s). Like in most disciplines, the longer it takes to discover a problem, the more expensive and time consuming it is to fix said problem. This does not change in Software Engineering, if anything, the stakes are even higher.

Algorithms Which Can be Used.

In agile development, it is often useful to have a visual representation of work left to do versus time left to accomplish said work. A very useful program to measure this is **Burndown Chart** and it is applicable to any project which has some quantifiable progress against a given time constraint.

Similar to a Cartesian graph, the pending or unfinished work is plotted on the vertical or y axis while the time period lies on the horizontal or x axis. It is quite an essential program for any team that wishes to accurately model and predict the time frame and window which will be used to complete a given task or project.

The team then adjusts the chart daily to determine how much work remains by summing the Sprint Backlog estimates every day of the Sprint. The amount of work remaining for a Sprint is the sum of the work remaining for the whole Sprint Backlog. Keep track of these sums by day and use them to create a graph that shows the work remaining over time. Burndown charts enables the team to:

- **i.) Monitoring the project scope creep.**
- **ii.) Keeping the team running on schedule.**
- **iii.) Comparing the planned work against the team progression.**

Another somewhat universal tool which can be used to record and access software metrics is **Github**. It provides a simple yet elegant solution for software developers regardless of whether or not they are working alone or as part of a unit. While using Github, each member of the team can work on the project simultaneously and then upload any and all changes or updates made to the project while having the original project saved to avoid potential errors.

This speeds up software development immensely while minimizing numerous code compatibility errors. Since each programmer can upload their, for the lack of a better word , work , Github makes it very easy for a team manager overlooking the project to make changes based on the work committed into the project's repository. The head of the team can easily see who is being productive based on not only the quantity of committed work per team member but also the quality of code and the regularity at which the work is being committed.

Using this information he or she can implement changes within the team to fix issues which arise or could harm the successful completion of the project, such as updating the schedule or changing code formats or even team compositions to make sure the project is carried out as efficiently as possible. Github also makes testing code a lot less bothersome since code can be committed to a branch without corrupting the Master version of the product. This means the team, if they chose so, could work on and test the code simultaneously, immensely speeding up the development process.

Accurate forecasts or timelines can then be constructed to give the developers and potential clients a reasonable time frame for the delivery of a finished product.

Consequences Of Measuring Software Engineering.

In any profession where workers and their products are measured by various metrics to find out their underlying worth or value so to speak, to the Company they work for or the project(s) they work on. It becomes quite easy to lose the anthropological aspect of business as everyday workers are reduced to points on a graphs or even numbers which more or less define them.

While measurement of software engineering metrics has the power to make or break Projects, the line between sufficient and invasive data is at best a blurry one if at all it even exists. For example some companies have begun regularly collect data such as respiratory rate and heart rate of their employees, at first glance this may seem innocent and even a possible safety measurement to catch symptoms of an illness at an early stage and maybe even prevent the sickness.

However, what if the company decides to use this data to factor into who gets a promotion or who works on future projects without the input of the workers. As time goes on more and more data will be collected from workers to ensure that the work environment desired by the employer is in place. Who decides what data is private and or personal and what data is up for grabs for the employer. Companies have advanced from simply collecting details such as blood type to now regularly logging details such as tone of voice used while working on the job. Where does it end?

In my opinion there is no simple answer, it’s far from being black and white and as with most things in the 21s century we must proceed with caution and hope for the best.

SOURCES :

What is software Engineering :

https://www.guru99.com/what-is-software-engineering.html

Software Measurement Metrics :

https://www.tutorialspoint.com/software_quality_management/software_quality_measurement_metrics.htm#:~:text=It%20can%20be%20classified%20into,improve%20software%20development%20and%20maintenance.

https://medium.com/@infopulseglobal_9037/top-10-software-development-metrics-to-measure-productivity-bcc9051c4615

https://stackify.com/track-software-metrics/

Algorithms Used :

https://www.visual-paradigm.com/scrum/scrum-burndown-chart/