Trinity College Dublin

The University of Dublin

Measuring Software Engineering
CSU-3012

Table of Content

1. Introduction

In this report, I will address the forms in which software engineering is measured. I will also go over the different algorithms used for this measurement and the available platforms. Finally, I will discuss the ethical issues surrounding this practice.

2. Software Engineering

Software engineering is a detailed study of engineering to the design and development and maintenance of software.
Problems arise when a software generally exceeds timelines, budgets and reduced levels of quality. Software Engineering ensures that the application is built consistently, correctly on time and on budget requirements.
Measuring software engineering involves the usage of software metrics.[1]

2.1. Software Metrics

A software metric is a measure of software characteristics, which are quantifiable or countable. Software metrics are important for many reasons including measuring software performance, planning work items and measuring productivity.[2]

2.2. Why is software engineering measured?

A software engineering task needs to be calculated, tracked and evaluated in order to achieve efficient and successful production. This will allow development teams to recognise low productivity areas and more efficiently prioritize the resolution of problems.
Software development teams can thus use these software metrics to communicate the status of their project, pinpoint and address issues, and manage their workflow in a better way.
They also allow Project managers to more accurately estimate development time and costs and identify areas that can be improved upon.[2]

3. Choosing Software

3.1.Specifications of software metrics

For management teams, software metrics are great because they provide a fast way to track the progress of software, set targets and assess performance. Oversimplifying the development of software, however can distract software developers from objectives such as providing useful software and increasing customer satisfaction.

Therefore it becomes imperative to make measurement of software easy to collect and measuring or analysing doesn't have to be burdensome or something that gets in the way of coding. Thus software metrics should be :

- Simple
- Consistent
- Easy to calibrate
- Cost-effective[3]

3.2.Examples of software metrics

3.2.1. Code Churn

Code churn represents the number of lines of code that were modified, added or deleted in a specific period of time. If code churn increases, then it could be a sign that the project needs attention. [2]

Advantages

- Simple to measure and visualise
- Can be tracked individually for each programmer
- Easy to spot trends in code

Disadvantages

- Cannot filter out comments in code
- Commit count or lines of code does not necessarily mean code written is high quality or not
- Shouldn't be used as a sole software metric as it can encourage developers to add redundant code

3.2.2. Cycle time

Cycle time describes how long it takes to change the software system and implement that change in production. It is a measure of the elapsed time when work starts on an item until its ready for delivery. There are a number of benefits from reducing cycle times such as

- Competitive advantage from products reaching the market sooner
- Better understanding and management of the corporate cost structure

3.2.3. Team velocity

Team velocity tests how many software units a team completes in an iteration. This is more of an internal metric that should not be used to compare teams as their tasks could be entirely different.

Advantages

- Team performance can be easily monitored over time
- Product owner can do release planning using velocity and story points

Disadvantages

- Teams with no slack or "*idle*" time become overly focused on individual objectives thus hurting their overall velocity
- Teams should not expect instantaneous results and rather expect for them to mature and realize their full potential

3.2.4. Production

Production metrics attempt to measure how much work is done and determine the efficiency of software development teams. The metrics that use speed as a primary factor are favoured by managers who want their products out as fast as possible.

Advantages

- Improved performance of workforce
- Transparent metric
- Balanced and proactive approach

Disadvantages

- Employee resistance
- Developer's metrics may not always be a true result of the final product

3.2.5. Agile process metrics

Agile process metrics focus on how teams using the agile software development method makes decisions. This metric does not describe the software and can only be used to improve the development process. A common metric used is called basecamp which I will discuss more about later.[4]

Advantages

- Testing is integrated throughout the project's lifecycle, thus ensuring quality of final product
- Brings flexibility to the project
- Possible to identify issues in early development as small releases are visible to client

Disadvantages

- No finite end as minimal planning is required and projects can tend to deviate
- Limited documentation

All the above mentioned metrics have their pros and cons and can be manipulated to give false results. Therefore companies should use a combination of metrics to have an accurate result.

4. Algorithms Used

4.1.Functional point analysis

Functional point analysis is a technique used to measure software requirements based on the different functions that the requirement can be split into.

Developed in 1979 by Allan J. Albrecht, he defined it as *"a dimensionless number defined in function points which we have found to be an effective relative measure of function value delivered to our customers"*.[6]

The FPA is a method of functional size measurement. It assesses the functionality delivered to its users, based on the user's external view of the functional requirements.

It is a tool to measure the units of a software product to support quality as well as productivity analysis. It can also be used as a normalization factor for software comparison.

4.2.Machine learning

Semmle.Inc used a machine learning approach to measure software development productivity. Semmle analysed over 10 million commits submitted by 300 thousand developers in over 56 thousand open source projects. The programming language QL was used to analyse the quality of the code.[7]

Their attempt was to measure the quantity of code produced by the amount of work time needed to generate the code. They trained the neural network to predict the likelihood that the developer was actually coding

during that time for every minute a specific developer spent between two different commits.

5. Platforms Available

5.1. Basecamp

Basecamp is a real time communication tool that developmental teams can use for resource planning and long term scheduling. Basecamp includes an in-built calendar, a to-do list and a system for file sharing which teams can use to keep a track on priorities.

Basecamp is not specific to a single industry, instead it can be used by any organization that needs to manage a group, including non-profits, start-ups and freelancers.[8]

5.2. GitHub

GitHub has an inbuilt code review tool in its pull requests. The code review tool is bundled with GitHub's core service.

GitHub allows a reviewer with access to the code repository to assign themselves to a pull request and complete a review. In addition to the pull request one is able to check history of changes and add comments.

The GitHub code review tool is a very handy one especially as it doesn't need any extra configurations. The primary issue with it is that it only supports git repositories on GitHub. A similar code review tool that can be used is GitLab.

5.3. Toggl

Toggl is a commercial time tracking tool that development teams can use to track time spent by developers on projects. It provides a dashboard which provides several data visualization tools that help in tracking progress and making the data look much cleaner.[9]

Toggle has several more features such as detecting idle time for users, summary reports which can be exported and integration with other tools like Asana and Basecamp.

6. Ethical Concerns

6.1. Data privacy and GDPR laws

As we discussed previously measuring the performance of software development teams using metrics require a lot of data. The types of data collected and analysed by teams may in certain cases, breach the privacy of

the developer. Therefore it is crucial that managers are fully transparent with their process and make it clear to the developers what kind of indicators are being used and what information is being monitored.

6.2.Possible incorrect use of metrics

We have several examples of measuring software engineering metrics(as discussed in section 3) for development teams to use. However they are needed to be used properly. For example if a team solely uses lines of code written as their metric, a developer who writes a less wordy but more efficient code will be marked incorrectly as opposed to one who writes a less efficient code but has more lines.
Therefore it is important that a combination of software metrics be used as well as the platform used should measure fairly.

6.3. Confidentiality of Programmers

In Singer's Ethical issues in empirical studies of software engineering, there is an example of a metrics researcher who examines the relationship between metrics and defect rates to determine which metric relates to software quality. He found that some programmer's source code had more defects than any other's. In his report he refrains from revealing the programmer's identity or any individual data however concerns arise when the artifacts in code can identify the individuals who created them. This raises the issues of consent, confidentiality and beneficence in regards to those individuals.[10]

7. Summary

In this report I have discussed why measuring the software engineering process is important for development teams. I have also outlines various methods, platforms and algorithms used for this process. Finally I raised some of the ethical concerns associated with this area.

8. References

[1]https://economictimes.indiatimes.com/definition/Software-engineering
[2]https://stackify.com/track-software-metrics/#:~:text=Metrics%20are%20an%20important%20component,to%20foster%20better%20team%20productivity.
[3]https://martinfowler.com/articles/useOfMetrics.html
[4] https://insights.sei.cmu.edu/sei_blog/2014/09/agile-metrics-seven-categories.html

[5]http://ptgmedia.pearsoncmg.com/images/0131424602/samplechapter/0131424602_ch05.pdf
[6] https://www.geeksforgeeks.org/software-engineering-functional-point-fp-analysis/
[7] https://semmle.com/assets/papers/measuring-software-development.pdf
[8]https://basecamp.com
[9] https://biz30.timedoctor.com/toggl-review/
[10] Singer, Janice & Vinson, Norman. (2003). Ethical Issues in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering. 28. 10.1109/TSE.2002.1158289.