Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Module: CSU33012 Software Engineering
Date: 19/11/2021
Name: Sarah Dolan
Student number: 193332721

Measuring Software Engineering Report

Abstract: This report considers the ways in which the software engineering process can be measured and assessed in terms of measurable data, an overview of the computational platforms available to perform this work, the algorithmic approaches available and ethical concerns surrounding these types of analytics.

Contents

1. Introduction

This report is split into 4 sections. The first section details how one can measure engineering activity. The second component consists of the platforms on which one can gather and perform calculations over datasets. The third section will be concerned with various kinds of computation that could be done over software engineering data, in order to profile the performance of software engineers.Lastly the final section will consist of ethical/ legal and moral concerns surrounding the processing of this kind of personal data.

2. What is Software Engineering

Software engineering is the process of analysing, designing, developing and testing software systems. Software engineering is a disciplined, structured method of programming that is used in engineering to improve the quality, time, and budget efficiency of software development. According to IEEE definition Software engineering is the application of a systematic,disciplined,quantifiable approach to the development,operation and maintenance of software; that is, the application of engineering to software. Similarly, a German computer scientist, Fritz Bauer, defines software engineering as the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines.

3. Measurable Data

The measuring of software engineering is a difficult task. Despite the large amount of data available, its accuracy, utility, and relevancy might be questioned. Software engineering measurement, it has been said, cannot be done effectively and so should not be done at all. The development process as a whole includes so many variations that choosing a single measure, or even a combination of measures, that successfully examines the process, is quite challenging. Although there is no single metric that can be used to evaluate software engineering as a whole, there seem to be numerous metrics that can be used to identify inefficiencies, save cost and resources, and highlight areas where productivity needs to be improved, so measuring software engineering is an important task that should be completed. To be more specific this report will investigate:

- Lines of code
- Lead time
- Code churn
- Number of commits
- Code coverage

Lines of code

A line of code (LOC) is any line of text in a code that is not a comment or a blank line in any case of the number of statements or fragments of statements on the line. LOC clearly consists of all lines containing program header files, declarations of any variables and executable non executable statements. Lines of code are only useful for comparing or estimating projects that are written in the same language and adhere to the same coding standards.Some advantages include its most useful metric in cost estimation, its alternates have many problems as compared to this metric, and it is very easy in estimating the efforts. Nevertheless, some disadvantages include increased productivity while raising more lines of code, it produces less efficient code, and it does not consider the complexity of code.

“Measuring programming process by lines of code is like measuring aircraft building progress by weight”
-Bill Gates

Lead time

The Lead Time is the time it takes from when a customer makes a request to when they receive it. Ie. the time taken between when a task is started and when it is completed. It is an excellent predictor of a team's ability to provide value to its consumers. It is a good indicator of how long it will take to complete tasks. A one-time measurement is limited in its use because it does not necessarily provide valuable information. The issue with lead time alone is that many teams don't pay attention to the factors that go into it. A task's lead time covers the time it takes to create it, define it, work on it, test it, and release it. It's difficult to find out how to improve something if you can't disassemble it.

Code churn

When an engineer rewrites their own code in a short amount of time, this is referred to as code churn. This software engineering metric evaluates a developer's ability to alter code. It accomplishes so by counting the number of modified, altered, or deleted source lines of code. It's usually expressed in lines of code.The key objective of measuring code churn is to enable

management control over the software development process by using it as an indicator. When there is a spike in the churn code, it may indicate that something is wrong with the development process, and management should assess the software engineering process. Reasons why the code churn may spike can include : unclear requirements, indecisive stakeholders, difficult problems, under engagement, prototyping, and polishing.

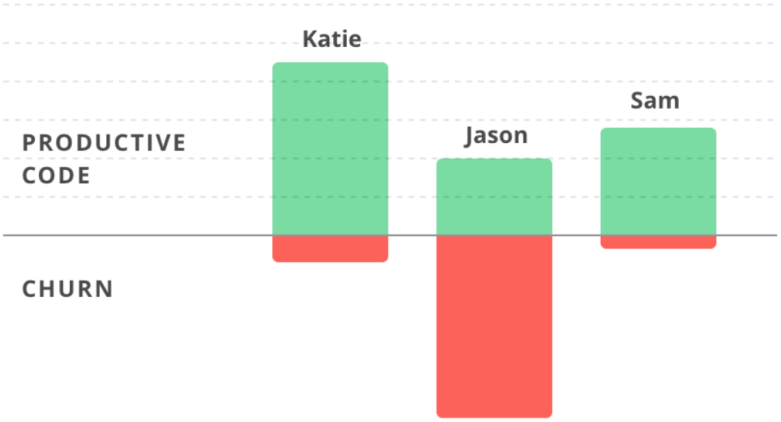


Figure1: an example of a code churn graph.

Looking at the productive code alone, it seems as though Jason has fallen behind in terms of coding the same amount as his fellow co- workers. Nonetheless, when looking at code churn we can see Jason wrote a lot of code, much of which was not used.

Number of commits

Engineers can also be judged based on the amount and frequency of their commits. Commits are arbitrary changes captured in a single moment of time.This appears to be a simple and efficient method of monitoring software engineering, yet it is ineffectual since commits do not correspond to accomplishments. Of course, it is good practice for programmers to commit on a continuous basis; nevertheless, what has to be committed is based on the developer's own preferences, rendering this an unproductive way of comparing developers. There are no defined criteria for how big or small a commit should be, thus it can be anything from changing the name of a variable to adding a comment to creating new functions or refactoring the code. In light of this, determining productivity based on the number of commits is ineffective, flawed and an arbitrary process.

Code Coverage

Code coverage is a metric that can help you understand how much of your source is tested. Code coverage is a percentage of the number of lines executed in the program as a percentage of the entire program. Testing whether each boolean case has been checked, if every statement in the code has been tested, and so on are examples of coverage criteria. Incomplete code coverage lowers a project's quality assurance.

4. Computational Platforms for Software Metrics.

There is an ever-growing range of platforms that not only facilitate but also specialize in the distribution of metrics. There are various new and innovative ways to do it, such as using an ergonomic piece of hardware to track when coding, designing, or otherwise. Some of these platforms include:

Codacy

While mentioned on their website, Codacy is an analytic software that evaluates the quality of work as it is being generated (Codacy, 2021). The site explains how to utilize it with any git-based service, such as GitHub, BitBucket, or GitLab. Code coverage, lines of code, errors, and the security of the code are all examples of code analysis it performs. This platform makes it simple to assess the quality of code written and determine the scope of any unit test design.

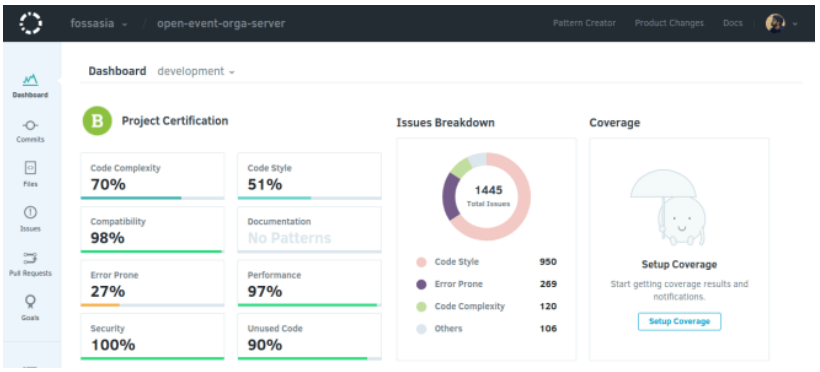
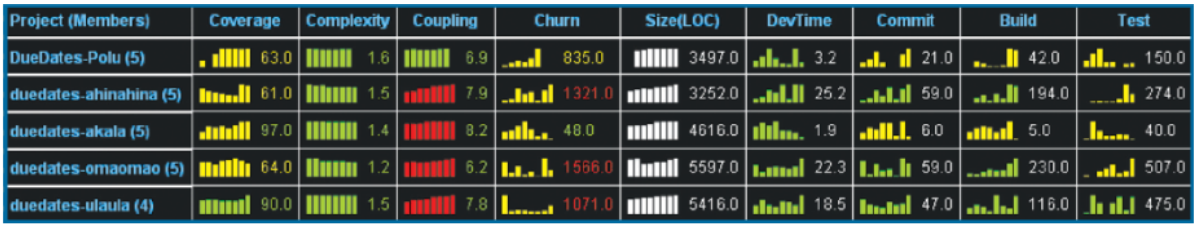


Figure 2: example of evaluation of code on Codacy.

Hackystat

Hackystat is an open source platform for gathering, analyzing, visualizing, interpreting, annotating, and disseminating information on the software development process and products. Hackystat users often attach software

'sensors' to their development tools, which capture and communicate "raw" data about development to the Hackystat SensorBase for storage in an unobtrusive manner.
Other online services can query the SensorBase repository to create higher-level abstractions of the raw data, combine it with other internet-based communication or coordination methods, and/or generate raw data visualizations, abstractions, or annotations.



Project (Members)	Coverage	Complexity	Coupling	Churn	Size(LOC)	DevTime	Commit	Build	Test
DueDates-Polu (5)	63.0	1.6	6.9	835.0	3497.0	3.2	21.0	42.0	150.0
duedates-ahinahina (5)	61.0	1.5	7.9	1321.0	3252.0	25.2	59.0	194.0	274.0
duedates-akala (5)	97.0	1.4	8.2	48.0	4616.0	1.9	6.0	5.0	40.0
duedates-omaomao (5)	64.0	1.2	6.2	1566.0	5597.0	22.3	59.0	230.0	507.0
duedates-ulaula (4)	90.0	1.5	7.8	1071.0	5416.0	18.5	47.0	116.0	475.0

Figure 3: example of computation approach form hackystat

GitPrime

Git prime is a project management tool that gathers data from the team's source control. It collects data and generates reports and insights on it. It runs on three different platforms. The development platform displays data such as the amount of commits and ticket activity, as well as graphs depicting how much the codebase has been impacted. It displays individual developers' activities so that they and their managers may see what they've done. The number of pull requests, commits, comments, and so on are displayed on the review platform. It is used to find inefficiencies in the work process, such as unneeded pull requests or code review disagreements. Over time, patterns emerge regarding how the code review process operates. The collaborative platform demonstrates how a technical team is cooperating. It is shown if one individual has a clear knowledge foundation for the team, and management may opt to train the team or team up other members to share information more evenly. It reveals which team members collaborate with one another and which do not.

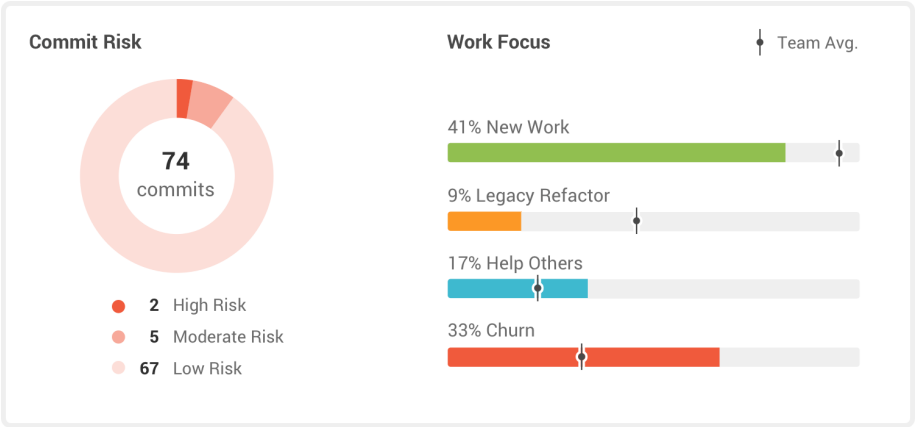


Figure 4: example of output of gitPrime output.

5. Algorithmic approaches

When it comes to evaluating software engineering, collecting data is crucial, but 'raw' data is useless if we can't make sense of it. Algorithmic models come into play at this point.

Halstead Complexity Measures

Halstead complexity measures are a set of metrics that are used to assess the complexity of software by determining which tokens are operands and which tokens are operators in each line of code. The following information can be gathered:

n1= the number of distinct operators
n2 = the number of distinct operands
N1 = the total number of operators
N2 = the total number of operands

The aforementioned figures can be used to compute program length, vocabulary (the total number of distinct operators and operands), volume (the amount of storage space required in bits), complexity (how tough the program is to write and understand in a code review), and effort (Time required). Halstead metrics have a number of advantages, including being simple to calculate, being able to anticipate the rate of error and maintenance effort, and being compatible with any programming language. The metrics, on the other hand, are based on completed code and are not widely utilized since they are perceived to be erroneous due to the lack of a precise method for identifying operators and operands, which might result in various calculation results for the same code.

Bayesian Belief Networks

A Bayesian Belief Network is a graphical depiction of various probabilistic interactions between random variables in a collection. It's a condition-independent classifier that doesn't rely on attributes. The probability in a Bayesian Belief Network is derived based on a condition — P(attribute/parent), i.e. the probability of an attribute being true over the parent attribute. It is a useful tool to visualize the probabilistic model for a domain, review relationships between variables and provide reason to casual probabilities within given scenarios. The advantage of these networks is that

they can not only compact and factor an exponentially huge distribution, but also supply this information to inference algorithms that can answer questions about these probabilities distributions without having to design them directly.

Computational Intelligence

The theory, design, use, and development of physiologically and linguistically motivated computational paradigms is known as computational intelligence (CI). Neural Networks, Fuzzy Systems, and Evolutionary Computation have traditionally been the three fundamental foundations of CI. Given input it can be used to generate feedback and make decisions. CI helps computers to comprehend problems that cannot be translated into binary order, so it is essentially a technique of functioning like humans. Neural networks, fuzzy systems, and evolutionary computation are the three primary foundations of CI.

Neural Networks- Artificial neural networks (NNs) are massively parallel distributed networks that can learn and generalize from examples, and they are inspired by the human brain. Feedforward NNs, recurrent NNs, self-organizing NNs, deep learning, convolutional neural networks, and other types of neural networks are studied in this area. They are used for data analysis and classification and therefore are extremely useful in examining data related to the software development process.

Fuzzy Systems- Fuzzy systems (FS) model linguistic imprecision and handle uncertain situations based on a generalization of traditional logic, which allows us to do approximate thinking, using human language as a source of inspiration.

Evolutionary Computation- Evolutionary computation (EC) addresses optimization problems by generating, evaluating, and altering a population of viable solutions, which is inspired by biological evolution.

6. Ethics

Measuring software engineering necessitates some sort of surveillance, which introduces ethics and legality into the equation. Organizations reverse the right to monitor their work. Where do we draw the line, and how do we keep data from being misused? Monitoring workplace efficiency and effectiveness will aid in the improvement of the qualities listed in the aforementioned

qualities. What's the difference between surveillance and stalking? Do employees understand how they're being observed and are they happy with the level of surveillance?

A couple of these questions are answered by Irish law. You must be informed about monitoring in Ireland, including who is watching you, what they are monitoring, how they are monitoring you, and when they are monitoring you. If your employer wants to monitor your Internet or email usage, it must be essential, ethical, and reasonable to the risk of you doing anything inappropriate using the Internet, such as leaking confidential information.

I believe this ethical issue goes hand and hand with GDPR (general Data Protection Regulation). GDPR was introduced into the EU(European Union) in May 2018. This was put in place to protect the employee's data. The introduction of these regulations means that employees have the right to information regarding the collection and processing of personal data, access to that data, have their data erased and question the data control about collecting their data if they consider it unlawful. As a result of this, the question of what data is necessary to store?, Is brought to light.

7. Conclusion

On close analysis and appraisal, the software development process, as well as methods for measuring and improving it, have been examined in depth in this paper. Software development is still a young industry, but it has evolved at a rapid pace since its inception. The industry has praised agile and flexible development procedures in recent years, drawing influence from other production-focused sectors that have strictly regimented developmental processes such as the waterfall method. I've discovered that evaluating and quantifying software engineering is far more difficult and complicated than one might think. Evidently, from the findings above there are numerous evaluation methodologies and platforms available, each with its own collection of pros and cons, and identifying one that maximizes the benefit to your company is imperative. While these software measurements may be beneficial, I've already touched on the ethical issues surrounding their collection and use. With all of this in mind, there is still potential for improvement, and in this ever changing industry, I have no doubt that the practice will become far more optimized and effective in the years ahead.

8. References

Tech Target. (November 2016). Software Engineering. [online]. Retrieved form Tech Target: https://whatis.techtarget.com/definition/software-engineering [Accessed 19 December 2021]

Tutorials Point (2021). Software Engineering Overview. Retrieved from Tutorials Point: https://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm [Accessed 19 December 2021]

Pluralsight. (November 16, 2016). Why Code Churn Matters.[online] Retrieved from Pluralsight: https://www.pluralsight.com/blog/teams/why-code-churn-matters [Accessed 19 December 2021]

Pluralsight. (November 17, 2016). 6 Causes of code Churn and What You Should Do About Them. Retrieved from Pluralsight:[online] Retrieved from: https://www.pluralsight.com/blog/teams/6-causes-of-code-churn-and-what-you-should-do-about-them [Accessed 19 December 2021]

Agile Academy (2020). Lead time vs Cycle Time. Retrieved from Agile Academy: [online] Retrieved from: https://www.agile-academy.com/en/agile-dictionary/lead-time-vs-cycle-time/ [Accessed 19 December 2021]

Manning Publications Co. (2021). Lead Time.[online] Retrieved from: https://livebook.manning.com/concept/software-development/lead-time [Accessed 19 December 2021]

Citizensinformation.ie (June 15, 2021) Surveillance in the workplace.[online] Available at: https://www.citizensinformation.ie/en/employment/employment_rights_and_conditions/data_protection_at_work/surveillance_of_electronic_communications_in_the_workplace.html [Accessed 19 December 2021]

Geeksforgeeks.org (April 26, 2021). Lines of Code (LOC) in Software Engineering. [online] Available at: https://www.geeksforgeeks.org/lines-of-code-loc-in-software-engineering/ [Accessed 19 December 2021]

Hackystat, n.d. A framework for analysis of software development process and product data. [online] Available at:[online] https://hackystat.github.io/ [Accessed 19 December 2021]

Codacy.com(2021):[online] www.codacy.com [Accessed 19 December 2021]

gitprime.com(2021):[online] www.gitprime.com [Accessed 19 December 2021]

Geeksforgeeks.org (August 5, 2020). Halstead's software metrics.[online] Available at: https://www.geeksforgeeks.org/software-engineering-halsteads-software-metrics/ [Accessed 19 December 2021]

Dr. Saed Sayad University of Toronto (2010). [online] Retrieved at: http://www.saedsayad.com/docs/Bayesian_Belief_Network.pdf [Accessed 19 December 2021]

Cis.ieee.org(2021) What is Computational Intelligence? [online] Available at: https://cis.ieee.org/about/what-is-ci#:~:text=Computational%20Intelligence%20(CI)%20is%20the,Fuzzy%20Systems%20and%20Evolutionary%20Computation. [Accessed 19 December 2021]

DataProtection.ie(2020) -GDPR - Data Protection Commission - Ireland. [online] Available at: https://www.dataprotection.ie/docs/GDPR/1623.htm [Accessed 19 December 2021]