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

CSU33012 SOFTWARE ENGINEERING

Measuring Software Engineering Report

Name: Jiss Joseph

Student Number: 18320724

Introduction

Technology is advancing at a rapid pace. Since software engineers are at the forefront of such development, it raises the question of how a tech company can analyse and keep track of an engineer's productivity.

The characteristic of software measurement is an expression of the size, quality and dimension of the product or process. There are two types of software measurement, direct and indirect measurement. Direct measurement is when a process or product is measured directly using the standard scale. Indirect measurement is when a process or product is measured by using parameters like the use of reference. When looking at the concept of Measuring Software Engineering there needs to be an understanding of how one can measure and keep track of the productivity of an engineer.

In this report, I will look at the different methods of how software engineering activity can be measured, platforms one can gather and perform calculation over such data, computations that can be done over software engineering data as well as the ethical issues that may arise from such analytics.

Software Metrics

Software Metrics is the measurement of various aspects of the code that is written. The measurement can include the quality, amount and progress made through developing a software. Not only is software metrics important for measuring productivity it also allows measurement of software performance.

Software metrics offer some benefits. It allows to keep track of product performance, improve the quality of existing software, or even predict how the software will be at the end of development. Software metrics are used in different ways. For example, managers use metrics to find issues in order to prioritize them according to its importance as well as communicate these issues. This ensures that the team is productive. Software developers use metrics to keep track of project development, find and fix issues as they arise, monitor, and improve the workflow for maximum productivity. In both the case of managers and software engineers, priority is given to productivity which in turn increases efficiency.

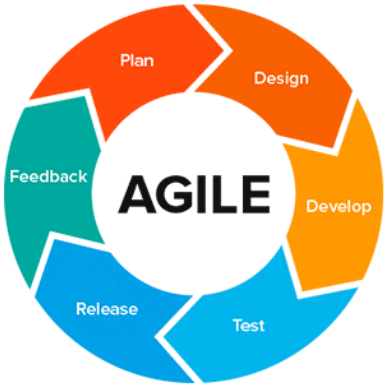
There are many different types of software metrics to measure the productivity of an engineer. Some of which I will discuss include:

- Agile Process Metrics
- Code Churn
- Function-Oriented Metrics

Agile Process Metrics

Agile development is a development process where teams work together to deliver their project to the customer in small increments. This kind of development ensures that the project is developed as fast as possible. It allows developers to adjust if the requirements given by

the customer change as the project goals and requirements are constantly evaluated. Some of the metrics that are used in agile process metrics are lead time, cycle time, team velocity and open/close rates.

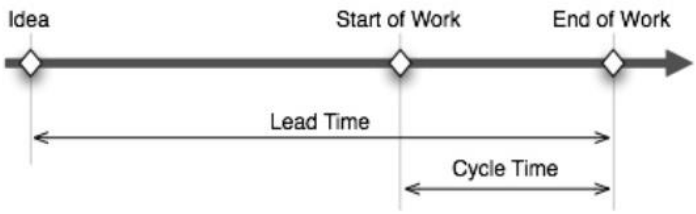


Lead Time

Lead time is the time that is estimated for the design plans to be developed as shown in the agile development diagram above. This ensures that engineers try the best to achieve this time limit in order to maximise the development process of each design idea. If possible, reducing the lead time given to an engineer, can aid in finishing the project faster, so that they can deliver the finished product to their client. This shows how productivity can also be increased.

Cycle Time

Cycle time is the time measured from when the work was started to when it was delivered to the client. Cycle time is another way in which productivity can be increased. If the cycle time is shorter then this shows the process was optimised and product can be delivered to client faster. Meanwhile, longer cycle times indicate that it will take longer for the final software product to be delivered to the client which in turn shows that there is some inefficiency within the development process. Jira is a project management tool that works well in keeping track of cycle time.(Will be explained later in the Platforms section of the report).



Team Velocity

Team Velocity is the amount of work that the team completes after each sprint. Reviewing the team velocity can give the owner of the project an understanding of how long it will take the team members to work through the backlog. This can be done as the report tracks multiple iterations of the completed work. Team velocity can be used in many ways but monitoring this velocity over time is important. For example, in newly build software engineering teams, you would expect team velocity to increase over time as developers familiarise themselves with teamwork patterns and workflow as well as team strengths. However, with existing teams, you can monitor the changes in velocity as well as ensure that it is consistent throughout the development of the project. Monitoring the velocity is important as it allows to track any inefficiencies. A decrease in velocity would indicate to the manager that the development process has become inefficient, and this will need to be corrected before the start of the next sprint.

Open/Close Rates

Open and Closed rates are calculated by keeping track of the issues that arise during the development process. It takes into account how many of these production issues were reported and resolved/closed within a certain period of time. Open and closing rates can provide an insight into development as well. For example, a high open rate and low closing rate, through multiple sprints may suggest that solving production issues are a lower priority in the development process or some other reason. This shows that this metric provides a brief insight into the development process. Although the root problem still needs to be tackled, this metric provides a starting point to monitoring productivity within agile development.

Code Churn

Code Churn is the number of the lines of code that were newly added, deleted or changed within a given time frame. A sudden increase in code churn outside the expected normal churn for that company can show that there are issues within the development process. The code churn level can vary depending on the project timeline. It can be used to monitor the productivity of a software engineer by indicating how well they are tackling issues they face within the development. It is a very useful metric signal for software engineering managers, and it can also improve team performance as a whole.

Function-oriented Metrics

Function-oriented metrics look at how much functionality the software provides. However, like it sounds, it is not possible to measure the functionality of a software directly. Therefore, the function-oriented metrics rely on the calculation of the function point. This function point calculation is a unit of measurement that enumerates the business functionality provided by the product. When software projects are written in different programming languages, function

points are useful in comparing these projects. The following steps can be used to calculate the function point:

- Step 1: Find the scale. It will vary according to the character of Complexity Adjustment Factor (CAF). i.e., F = 14 * scale
- Step 2: Calculate the Complexity Adjustment Factor. i.e., CAF = 0.65 + (0.01 * F).
- Step 3: Calculate Unadjusted Function Point (UFP).
- Step 4: Calculate the function point. i.e., FP = UFP * CAF

There are both advantages and disadvantages to Function Points. Looking at the advantages, it works for different programming languages, and it is useful for software analysis. The disadvantage with Function Points is that the concept is difficult to understand, and the methods used change. This is one of the reasons why many software engineers skip using this metric.

Platforms that enable Gathering and Processing of Data

GitHub

GitHub is a collaborative tool that allows one to manage their projects. It provides internet hosting for software development and acts as a version control using git.

For my visualisation project, GitHub proved to be a very useful tool in keeping track of different versions of the code. It allowed me to revert to older versions of the code that were working if the newer version had issues. As such GitHub offers a lot of features that increase software development productivity.

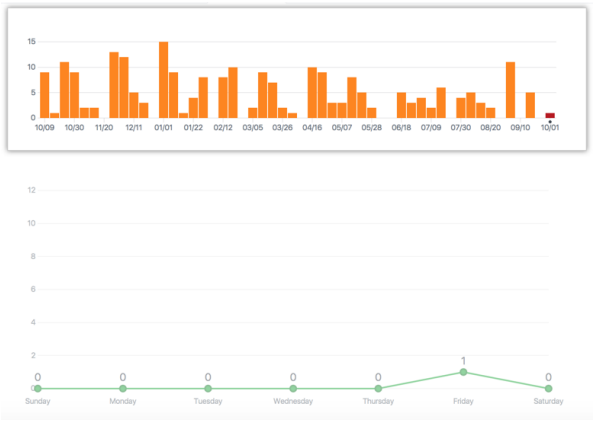
GitHub Features:

- Collaborative Coding
    - You are able to contribute to projects quickly with automatic environment setup.
    - Pull requests - Pull requests allow other developers in the project what changes that you have made and pushed onto the branch in a repository. A pull request can be reviewed by other collaborators to see the changes and make further changes before the code is merged with the base branch.
    - Count of total commits as well as commits of individuals.
- Project Management
    - Project boards – Help with organising and prioritising work. It allows you to set customized workflow that suits project need and specifications.
    - Issues – Track bugs, enhancements, prioritize work.
    - Unified Contribution Graph - Shows all the contributions that you have made on GitHub. This can be viewed as a contribution graph.
    - Organisation Activity Graph – Visualises all the data collected from your contributions to projects. Organization Activity insights offers data visualisation of the entire organization of specific repositories, with issue and pull requests activity, the main languages that are used, and information on what team members spend their time working on.

- Repository insights – Use data activity and contributions within repositories including trends, to make data-driven improvements to the development cycle.

Sample Code Frequency Graph

Sample Commits Graph



GitHub allows tech companies to track the work done by the software engineers. They are able to manage projects in an organised manner through using git. The data collected from the contribution of each person can be used to see their productivity within the team. For example, employers can see how long it took for an engineer to complete a certain feature/task/issue. This proves useful when making a start at measuring productivity. The task management tool within this GitHub platform allows for software metrics to be calculated from the data collected from its contributors. For example, team velocity, open/end rates and cycle-times to be calculated.

Jira

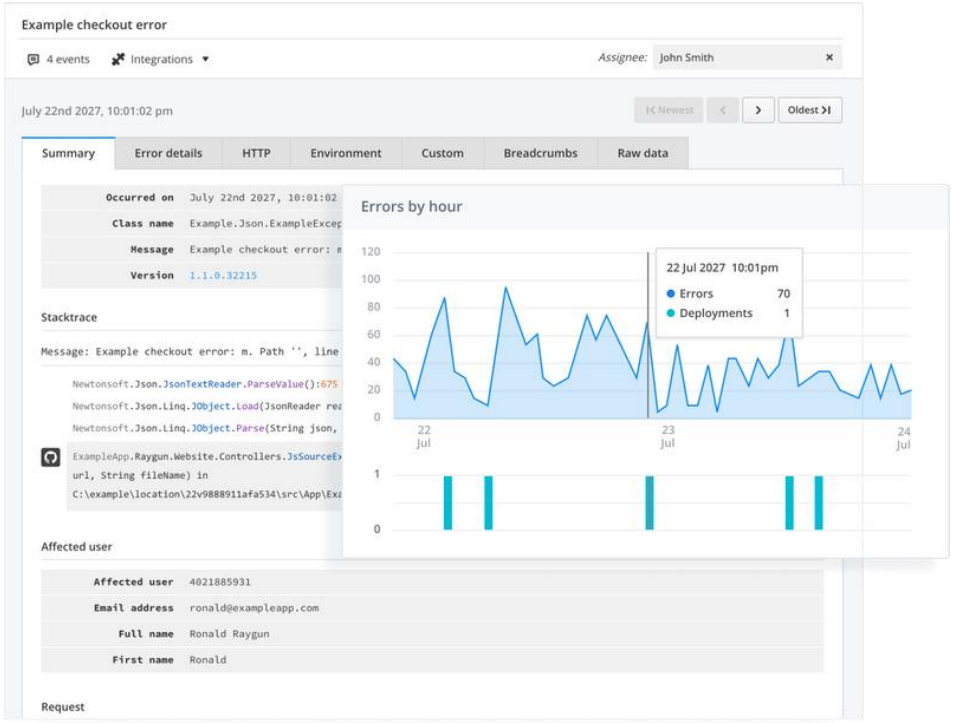
Jira was designed in the beginning as an issue/bug tracker. However today, Jira has become a powerful work management platform for agile software development as well as other use cases.
Jira offers many features that benefit agile development. Some of the features include scrum boards and Kanban boards. These are very useful for software engineers as they help manage and monitor work/issues within a team. It helps the teams to work more efficiently as they are all aware of the project objectives as well as what each other team member is working on. This sort of transparency can reduce the lead time of the project. Jira also offers time tracking capabilities such as team velocity reports, sprint reports and cycle times. Jira collects the data from the contributors and provides the aforementioned features which greatly improve the monitoring of contributions made by each team member. This can greatly help in identifying any weaknesses within the development process, which allows the issues to be fixed and in turn increase the team velocity.

Raygun

Raygun is a cloud-based platform that provides reports on what errors/bugs/crashes an application faces.

Raygun offers insights into the performance and quality of applications. This platform provides error monitoring and crash reports that allow engineers to resolve issues as they appear. This means that issues can be resolved faster, instead of the developers spending extra time trying to find the bug. This will greatly improve the experience of the end user. Raygun also provides real user monitoring and application performance monitoring. These are useful tools that help software engineers to solve front-end performance issues. The application performance monitoring similarly provides insight into code root causes of certain issues. The time saved on this allows engineers to resolve the bugs faster and provide a better user experience. Both these features, save developers time which they can then contribute to maybe implementing another feature etc. This increases productivity within the development process.

Raygun – Error Monitoring and Crash Report



Computation of Data and Algorithms

COCOMO (Constructive Cost Model)

The Constructive Cost Model is a procedural cost estimation model for software. It is based on a regression model LOC, which is the number of lines of code. It was used often to predict the different parameters which are linked to making a project. For example, the size, effort, cost, time and quality. This model was developed by Barry W. Boehm in the 1970s. The

model is based on the study of 63 projects, which makes it one of the best documented models.

The two factors that define the quality of a software product is Effort and Schedule. The Constructive Cost Model ensures both:

Effort – The amount of labour that is required to finish the work. This is measured in person months units.

Schedule – This defines the amount of time that is required to finish the work. This is proportional to the effort made. This is measured in units of time like weeks, months.

Different models of the Coscomo predict the cost estimation at different levels. This is determined by the amount of accuracy required. This model can be applied to various projects depending on the need and requirements.

There are three types of Cocomo model, with increasing accuracy. These models are the Basic Cocomo Model, Intermediate Cocomo Model and the Detailed Cocomo Model.

The first type/level of the model is the Basic Model, which is used for rough and quick calculations of the software costs. The accuracy of the software cost is restricted due to the insufficient factors.

Intermediate Cocomo

Intermediate Cocomo is finds the software developed effort as function of program size and a set of cost drivers that include assessment of product, personnel, hardware, and project attributes. Each of these attributes have their own subsequent attributes:

1. Product attributes have attributes of required software reliability extent, size of the application database and complexity of the product.
2. Hardware attributes have attributes of run-time performance constraints, memory constraints, volatility of the virtual machine environment and required turnabout time.
3. Personnel attributes have attributes of analyst capability, software engineering capability, applications experience, virtual machine experience, programming language experience.
4. Project attributes have attributes of use of software tools, application of software engineering methods, required development schedule.

The manager of the project is to rate the 15 attributes above, from very low to very high according to its importance. According to the rating of the cost driver, the equivalent values are used. The EAF(Effort Adjustment Factor) is then calculated by multiplying the above determined 15 values.

The Intermediate COCOMO formula is:

$$E = (a(KLOC)^b) * EAF$$

Depending on the software projects the values of a and b is taken.

Project	a	b
Organic	3.2	1.05
Semi Detached	3.0	1.12
Embedded	2.8	1.20

- E is the in person-months effort.
- KLOC is the estimated number of lines of code delivered.
- EAF is the Effort Adjustment Factor
- The a and b values are calculated from using the table above.

COCOMO is a useful calculation as it provides a cost estimate model for software projects.

Software Engineering – Halstead's Software Metrics

Halstead's Software Metrics are software metrics introduced by Maurice Howard Halstead in 1977. Halstead found that the metrics of the software program should reflect its implementation or expression of its algorithm in different languages, but independent of their execution on a specific platform.

It is calculated by using the following variables:

$$n1 = the\ number\ of\ distinct\ operators$$

$$n2 = the\ number\ of\ distinct\ operators$$

$$N1 = the\ total\ number\ of\ operators$$

$$N2 = the\ total\ number\ of\ operators$$

Using these variables, the following can be calculated:

- Program Vocabulary: $\mathrm{n} = \mathrm{n1} + \mathrm{n2}$
- Program Length: $\mathrm{N} = \mathrm{N1} + \mathrm{N2}$
- Estimated Program Length: $\mathrm{N} = \mathrm{n1}\log_2 \mathrm{n1} + \mathrm{n2}\log_2 \mathrm{n2}$
- Volume: $\mathrm{V} = \mathrm{N} * \log_2 \mathrm{n}$
- Effort: $\mathrm{E} = \mathrm{D} * \mathrm{V}$
- Time required to run program: $\mathrm{T} = \frac{\mathrm{E}}{18}\mathrm{seconds}$
- Number of Bugs: $\mathrm{B} = \frac{\mathrm{E}^{\frac{2}{3}}}{3000}$

The use of Halstead Metrics has both advantages and disadvantages. These are:

Advantages

- It is easy to compute.
- It measures overall quality of the program.
- Predicts the rate of error.
- Predicts maintenance effort.

- Does not require the entire analytics of the entire program structure.
- It can be used with any programming language.
- It can be used for managing projects i.e., Scheduling and reporting.

Disadvantages

- It is dependent on the code being fully finished.
- Does not function as an estimating model.

Halstead's Software Metrics had a positive impact in software development. It helped measure the performance of a software. Halstead's metrics demonstrated that the complexity of a given set of code is dependent on the abstraction level and volume of the software.

Ethics

It is seen in this report that the collection of data, through the explained software metrics is a vital part in measuring software engineering. It is seen that the productivity and the quality of the work done needs to be measured in order to accurately "measure" the contribution of an engineer. Not only do the metrics help in monitoring the contributions, but also help tackle issues that may arise within both the software development process as well as the business aspect. This raises issues of privacy, as data must be collected from employees in order to measure productivity.

The issue with employee privacy is prominent when measuring software engineering. The platforms used by tech companies like the ones I have explained above uses the data collected from the work completed to measure productivity. When data processing is done right, tracking the work done can have advantages for both the employer and employee. As said above there are many reasons why monitoring the work of software engineers is seen to be useful. Issues may arise where employees are unaware of the amount of data that is collected about them by the workplace. The extent of how much information is collected about employees from work, communication or non-work-related monitoring of data is what raises ethical issues. Companies have legal regulations and digital privacy to protect their employees but at an age where technological innovation is rapidly increasing, it has outpaced many of these legal regulations. Companies may collect data past the required data sets without employee knowledge and share them with the company, which is a breach in the employee's digital privacy. In the sector of software engineering monitoring and tracking the work is an important part in determining the productivity and work ethic of an engineer, so this set of monitoring is both useful and necessary. Tech companies need to determine these necessary aspects of data collection by assessing what measures the work of a software engineer and store this data securely for this purpose of assessment without sharing it with third parties.

Conclusion

In conclusion, there are many software metrics that are used to measure software engineering, some of which were discussed above. They are however a bit complicated when examining and determining the productivity of an engineer. Not only are software metrics a useful way of monitoring employee workflow, but it can also be used to improve the work ethic and productivity of the team overall. The use of software metrics is a necessary aspect of software development. It does however raise ethical issues of employee digital privacy. If data is collected for the sole cause of measuring software engineering and if these metrics and calculations are made clear by the employer, it can have a positive impact on both the employee and employer.

References

Introduction

https://www.geeksforgeeks.org/software-measurement-and-metrics/

Software Metrics.

https://www.educba.com/software-metrics/

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

https://linearb.io/cycle-time/

https://www.atlassian.com/agile/project-management/metrics

https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams

https://www.geeksforgeeks.org/software-engineering-calculation-of-function-point-fp/

https://www.totalmetrics.com/function-point-resources/what-are-function-points

Agile Development Image

https://medium.com/moodah-pos/agile-development-95cad3573abf

Cycle Time and Lead Time Image

https://getnave.com/blog/kanban-cycle-time/

Platforms that enable Gathering and Processing of Data

Github

https://github.com/

https://github.com/features

Github Images of Commits graph and Code Frequency Graph:

https://docs.github.com/en/repositories/viewing-activity-and-data-for-your-repository/analyzing-changes-to-a-repositorys-content

Jira

https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for#Jira-for-requirements-&-test-case-management

Raygun

https://raygun.com/

Raygun – Error Monitoring and Crash Report Image

https://raygun.com/?utm_source=generic&utm_medium=adwords&utm_campaign=brand&utm_term=raygun%20software&utm_matchtype=e&gclid=EAIaIQobChMIgLPK0uL39AIVmLPtCh22bgGlEAAYASAAEgKiT_D_BwE

Computation of Data and Algorithms

COCOMO(Constructive Cost Model)

https://en.wikipedia.org/wiki/COCOMO

https://www.geeksforgeeks.org/software-engineering-cocomo-model/

Halstead's Software Metrics

https://www.geeksforgeeks.org/software-engineering-halsteads-software-metrics/

https://en.wikipedia.org/wiki/Halstead_complexity_measures

Ethics

https://www.computerworld.com/article/3642712/rise-in-employee-monitoring-prompts-calls-for-new-rules-to-protect-workers.html