Measuring the Software Engineering Process
Conor Thompson - 17331651

Overview:

As stated in the manifesto found at https://www.scss.tcd.ie/Stephen.Barrett/teaching/CS3012/index.html , the purpose of this report is “to deliver a report that 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 the ethics concerns surrounding this kind of analytics”. In order to provide as focussed a report as possible, my subdivisions in this report will be based on the topics provided.

Why Measure the Software Engineering Process?:

The measurement and analysis of software engineering as process is a powerful tool in the arsenal of management staff in a firm. Thanks to its strength and versatility, it is a tool which must be looked at in depth to ensure that there are viable implementations of this tool which limit the potential for its abuse or misuse by members of management. These tools are held in high regard by those who use them as they streamline portions of their work by use of metrics about a multitude of aspects of any particular individual or team and by providing a means by which to rank said teams and engineers.

Methods of Measuring Software Engineering Output:

When applying measurements to the software engineering process, one of the most important decisions that a company must choose early on is whether to place more emphasis on measuring individual output or evaluating teams as an overall unit. The prevailing trend is to place a strong emphasis on individuals’ ability to perform as engineers in the workplace, with a large number of metrics being able to be accessed to enable management to assess how all of their employees fare on their own. Many executive level staff are more than happy to make use of this style of analysis, as it enables them not only to isolate the top n% of their workers so they can find suitable employees for promotions and bonuses, but also to find the bottom m% of employees, whom they can cull from the rankings in a bid to eliminate waste.

One method of evaluating individual engineers which I found interesting upon hearing about it is a sort of gamification of the analysis of engineers. Arista, for example, divide their staff into ‘leagues’: there are 6 leagues into which all employees are categorised, and your ranking relative to your peers is used as a basis upon which to determine whether or not you will be considered for a promotion, and also determines how large of a bonus one will receive. A couple of observations regarding this method include that it provides a streamlined method of spotting prospective senior staff for the current cohort of management, but that it also makes the whole aspect of having your every piece of work monitored feel a bit less overbearing. The league structure implemented in this company makes the whole process feel more like a game than a corporate tool that largely determines your career path and future, which could help to mitigate negative feelings that employees have about the level of scrutiny afforded to management for this task by executive staff.

Individual analysis on its own is not an entirely fair practice. It is impossible to establish an accurate and sensible view of how your team-oriented workplace is performing when you never measure how your teams overall are performing. Many of the prevailing methods for analysing engineers can be as easily applied to teams as well as individuals, and seeing as a project’s teams are the units which are actually contributing to the output of an end product rather than those team’s constituent members, it makes a great deal of sense to apply these same methods of measurement to teams as to individuals. Such methods include:

- Measurement of number of lines of code written
- Measurement of commit counts
- Analysis of number of bugs solved vs. number of bugs caused by a team
- Use of managerial reports by managerial-level staff on both teams and their members
- Productivity metrics e.g. active days, efficiency, assignment scope.
- Production metrics such as Mean Time Between Failures (MTBF) and Mean Time To Recover/Repair (MTTR), Application Crash Rate

Another aspect of team-based analysis that may be worth consideration by managers is that by neglecting either individual or team analysis, they may fail to gain a good understanding of the bigger picture of how their employees are performing. It might be observed that there are engineers on a team who do not seem to produce as great a deal of work, but upon closer inspection are in fact taking on more challenging tasks which take more time to complete but are more

significant than the trivial problems some of their more highly-rated peers might have undertaken.

Analysing the number of lines written is also a highly fallible metric for managers who do not understand the differences that can be seen across programming languages. An obvious example of this would be to compare the number of lines written by a Java programmer compared to a Haskell programmer. Due to the fact that Haskell is a far more succinct language than Java, it might seem to the manager who is unaware of the behavioural differences between programming languages that both engineers wrote programs with comparable levels of functionality but with a vast difference between line counts. If a technical manager were incompetent enough to not take the innate differences between the languages used by engineers in their projects, the Haskell engineer might receive a nasty surprise in being ranked as a weaker engineer than his Java counterpart.

Analysis of the number of commits provided by an engineer could lead the unaware manager to fall into a similar pit as the snare that lies in only measuring number of lines written: engineers who are tackling non-trivial problems may not be committing to a repo as regularly as those who are working on less challenging problems. This could also result in those who have thrown themselves at the less obvious-to-solve problems in a company finding it more difficult to prove themselves as being as competent as other employees who are tackling issues that don't require as much time to solve.

A more reasonable metric by which engineers could be compared is the measurement of the number of bugs solved by engineers in their commits compared to the number caused by their code in commits. A metric along these lines would enable a manager to observe whether or not any particular engineer is more of a liability than an asset. If it is found that, over time, a pattern emerges that shows that for some engineer, they, on average, cause more bugs to emerge in the product than they fix in their commits, then it may be found that it would worthwhile for management to consider a closer assessment of said engineer to see if it is worth keeping them onboard as a member of staff.

If a manager wants to gain a carefully put-together view of all their staff, they must do their best to balance the results yielded by any metrics they use to try and obtain an holistic view of all of their employees. Certain tools and measurements might paint some strong engineers as weak, but it should always be remembered that the big picture is the part to consider most strongly.

Productivity metrics are a valuable tool as they give management the ability to closely monitor how invested their staff are in their assigned projects, and thereby a manager could make sure that the engineers under their charge are adequately committed to their work and are actively working to provide as effective a product as possible.

Once a product has been released, production metrics become an invaluable tool for analysis of the efficacy of the implementation in release. The Application Crash Rate of a program allows one to gain a view of the big picture for the performance of an app. MTTR grants an insight into how well engineers are gaining control of bugs and user issues in an app. Having crashes near the start of release is not a scathing mark against the ability of engineers or of teams, and management must bear in mind that having crash reports after releasing a product can often be unavoidable; there are many bugs that can only be found when an app has a large user base.

Platforms Available for Assessing Software Engineering:

As measurement of the software engineering process is an ever-increasing demand from management and as an emerging market which is experiencing large-scale growth, there are a large number of competitors available on the market. Notable examples include:

- GitPrime
- Waydev
- Jasper
- Hackystat

GitPrime and Waydev are probably the most popular options of the ones mentioned above. Both platforms are highly comprehensive and serve to streamline the analysis process for management. They are accessible tools and can provide great utility with a comparatively small degree of effort as compared to rival platform.

According to the GitPrime website, their platform allows one to “identify bottlenecks, compare sprints, and releases over time”. Their key aim is to make it as easy as possible for management to access and read facts & figures regarding performance in their teams. The Work Log provided means that metrics such as code commits and tickets are in a single place. It is also possible to learn more about how engineers are performing on a day to day basis thanks to GitPrime. The

timespan for viewing activity can also be modified to suit the needs a manager might have at any particular time, with the ability to view overall activity on a daily, weekly, or per-sprint basis.

One compelling feature of GitPrime is the Retrospective feature, whereby one can see and measure how successful different releases were and spot differences between performance across sprints in an Agile development cycle. This tool allows managers to obtain more information about these sprints and releases so that they can make more informed decisions about how to approach future releases.

The Code Fundamentals component of the GitPrime service splits analysis into four key sections:

- Coding Days
  -> One Coding Day is a day in which an engineer contributed code

- Commits Per Day
  -> This measure the number of non-merged commits per day

- Impact
  -> This metric assesses the severity of any changes to the codebase

- Efficiency
  -> GitPrime Code Fundamentals Efficiency metric is a measure of the ratio of contributed to churned code. Churned code is defined by GitPrime as being a measure of the amount of code rewritten by an engineer in a short span of time.

Between these four sections, a large degree of information can be obtained about performance in a workplace. It is clear from this that GitPrime is quite powerful and that it provides a lot of scope for management to make profound insights in the operation of their workplace.

Waydev is a tool that is cornering the same market as GitPrime. Its most helpful feature is how visual it is and how easy to understand its information is: its interface is very graphic, which allows you to see patterns quickly and intuitively.

On the FAQ section of their website, their metrics are listed. These are divided into Impact and Commits/Day. Waydev defines Impact as being “a metric which shows

you how much “cognitive load” did the engineer carry when developing these metrics”. In order to calculate a developer’s Impact, they assign a value to all of their commits based on how much original content it provided and how useful it was, with factors such as churn being taken into consideration.

The Waydev Commits/Day metric which shows managers how many commits their teams are pushing per day. It is defined by Waydev as “the ratio between total commits pushed by the entire team and the number of days with git activity on your project”, and for the purposes of this calculation, a commit is reckoned to be an individual change to a file or files.

When looking at the features provided by Waydev, it can be seen that their focus is almost solely on gathering a customer base among executive- and management-level staff in a firm. Their primary intention is to amass data about a large number of employees and give both high-level and in-depth analyses of numerous aspects of these employees, individually and overall. There is not as much purpose to using the app as an individual engineer as the majority of its functionality will be meaningless to you.

Ethical Concerns:

In my own estimation, there are a couple of key ethical concerns which ought to be discussed more, these include:

- The fear of unauthorised data gathering, lack of consent to analytics details on analysis performed obfuscated in employee’s contract
- The potential for data obtained about an employee being accessed by a third party without permission or by management (potential for blackmail or leverage)
- The degree to which engineers are being measured by these platforms and the question of how much of this is actually necessary
- The inability to remove all traces of you data if you revoke consent with full confidence.

With regards to unauthorised data gathering, there are very real reasons for people to be fearful of this happening to them. In the wake of Cambridge Analytica and similar scandals, there has been a lot of much-needed discussion on the topic of personal privacy and the extent to which data points about them are being gathered and processed by both private and public bodies. Thanks to information

about such groups that has been leaked by numerous whistleblowers over the last few years, people can become more informed about what information can be gathered about them, the purposes of this gathering, what is done with information collected about them (targeted ads and surveys being notable examples of this), and the motives of the organisations that are assembling this level of information on them and many other people just like them.

One welcome development in the wake of all of this is the creation of the EU's General Data Protection Regulation (GDPR). According to the website eugdpr.org, the aim of GDPR is "to protect all EU citizens from privacy and data breaches in today's data-driven world". GDPR came into effect in May 2018 and provides a comprehensive set of regulations regarding what can and cannot be done with any EU citizen's data. One of the key features of GDPR is that the penalties applied on non-compliant organisations are fairly substantial: either 4% of global turnover or €20 million, depending on which is larger. This is the most severe punishment that can be issued, but there is a tiered system in effect, so minor breaches of conduct will still receive fines, they just won't be as large. All people who are in possession of user data must operate under the confines of GDPR guidelines, as according to the regulations they are considered data controllers.

It is right for people to maintain concerns about third-party access to their information. When a company gathers data on their engineers, they must be sure that they have reasonable guidelines in place to prevent hackers from gaining access to this information. There have been any number of data breaches in the recent past, and in a lot of the successful cases, a great deal of confidential and personal information has been made public.

When a company decides that they will use a set of metrics to analyse their engineers and the process they undergo to bring a product to completion, it should be ensured in the contract of each employee who will be assessed with these methods, that information regarding these metrics and the data analysis process should be made clear and easy to find. It would be highly unethical to issue a contract to employees and render elements that you'd prefer they don't know about too difficult to find. When taking on a new employee, you must be sure that they're aware of the extent of the data gathered on them, what's going to be done with it, and any other information that they may need to know. To do otherwise would be a tremendous breach of trust between an employer and an employee.

Upon leaving a position in a company that was analysing their performance, an engineer must be able to be certain that all instances of data about him, and all relative data points to his behaviour in that job can be removed from the company's database easily and effectively.

Conclusion:

To conclude, I believe that the different methods of analysing the software engineering process can provide tremendous benefit to both employer and employee. Management can receive great insight into how their staff and teams perform and operate, and engineers can also learn from receiving more relevant information in managerial reports. Thanks to the emerging platforms for handling this analysis, staff can choose a competitively-priced system that suits their exact needs and works for the development methods used by their teams.

There are of course privacy concerns surrounding all of this, but largely due to the numerous whistleblowers who have shed light on how institutions are handling deeply personal information, staff can now make a more informed choice about how they will approach the issue, and devise more relevant ways to mitigate concerns about how they handle data.

References:

1) https://ieeexplore.ieee.org/document/93686

2) https://www.gitprime.com/platform/code/

3) https://waydev.co/

4) https://blog.gitprime.com/why-code-churn-matters/

5) https://waydev.co/gitprime-vs-waydev-vs-code-climate/

6) https://eugdpr.org/

7) https://www.wired.com/amp-stories/cambridge-analytica-explainer/

8) http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=2101&context=theses

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

10) https://www.sciencedirect.com/science/article/pii/S0164121299000357