Fraser O'Byrne
17329693
CS3012

Measuring Software Engineering

Introduction

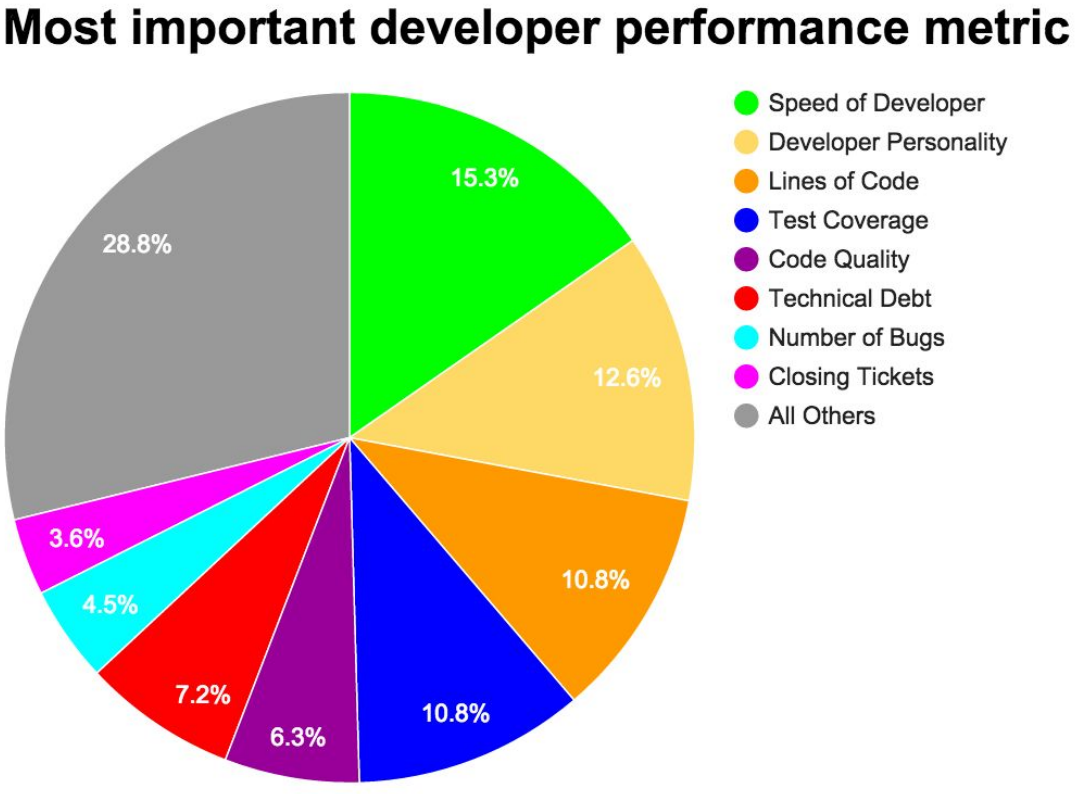
In this report I will discuss the different methods of measuring an engineers productivity, which aspects should be measured or prioritised and the ethical impact of all these methods, according to my own personal thoughts on the matter. The measurement of a software engineers productivity has been widely debated about which aspects to prioritise and how to perform this measuring ethically.

What is Software Engineering

Software engineering is one of the biggest growing careers in the world at the moment and does not look like it will stop growing for a good while longer. A software engineer create and manage applications that makes the experience of using an electrical device somewhat easier. They do this through testing and debugging. They strive to create different applications that can be used in not only their own project, but on many others so it has more than a singular use. Software engineers essentially take some application, improve it some bit and give it back out allowing other engineers to improve the already improved application.

Measuring a Software Engineer

The ability of being able to measure a software engineers performance and productivity is a subject of great debate. This debate includes how we should do it and which should they be measured on. Within the debate of which aspects should be prioritised, there is no clear favourite among developers. In a survey done by Brian York, when asked what, in their opinion, was the most important attribute to measure a software engineer, the response was very split.



This pie chart states the response of over 300 software engineers from the experiment. Many people state this chart is wrong as speed and personality are the two most important aspects when measuring the

performance of engineers. What we can take from this is that there is no set attribute that should be specified as to do so would isolate other aspects of an engineer that may make or break their productivity, or their ability to challenge for the title of a 10x. A 10x is a software engineer that can do the work of 10 engineers.
Software engineering cannot be measured purely on one or a few fields, instead the field should depend on your work environment and your project. For example using the personality trait above; yes, it would be important to have a good personality trait when working in a large team, yet it wouldn't be if you were in charge of a solo project. All the traits are circumstantial depending on what you're doing and how you are doing it.

Methods for Measuring Productivity

Sprints vs Burnouts:

One solution that would help deal with these varying factors is a method recommended by Ross Smith, the Head Architect of PITSS. He recommended using agile development practises, which is essentially sprints. These short sprints would be used to see how many sprints an engineer can take before inevitably burning out. This would be done over a two week period thus making it a time efficient way of checking how long productivity can stay. This would allow managers to effectively schedule deadlines to adapt to these burnouts so as the engineers will come close to burning out but not quite getting there. Thought an apparent fault to this method is how effective the sprints are. If we are looking for global standard way to measure the productivity of software engineers, can we use the same sprint samples for ever project out there? The obvious answer is no. This

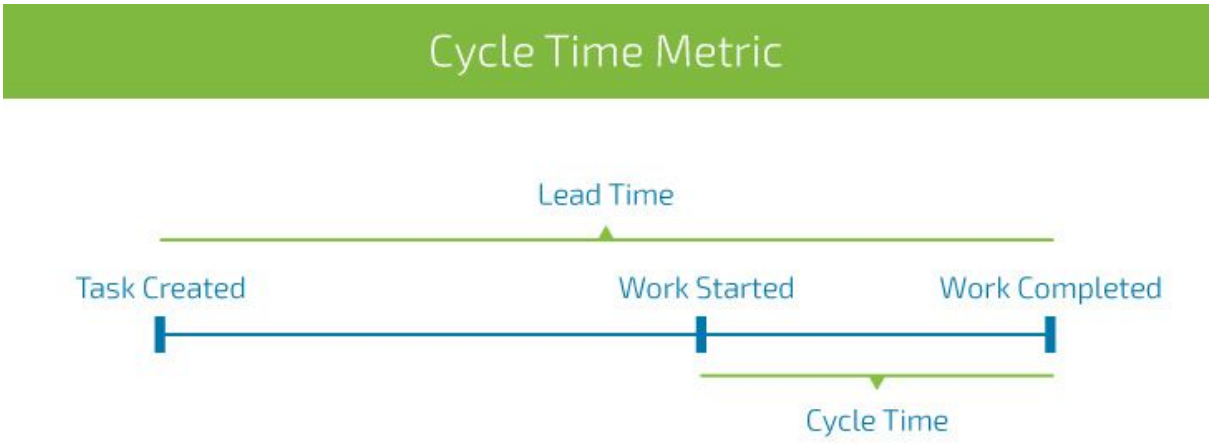
method is much more suited to an enclosed community, where if used elsewhere the sprint task would be required to change.

Economic Impact:

Another example of measuring productivity is the overall impact a project makes on its budget once finished and ready for the world to use. This is quite a standard way of measuring the performance of a product as it is just simple calculations and requires not much statistical analysis. This method is only useful once after the project so cannot show to productivity during the project, therefore losing out on efficiency compared to other methods that can analyze efficiency during the project.

Cycle Time:

Cycle time is described as the time elapsed between the start of a certain feature to the time the feature is delivered. This allows a team to see how quickly a feature of a project can be completed and base it against other features in the project, thus having some set standard to where their productivity should be.



This table shows exactly how this works within a timeline. This method, though, has its downfall where different features take different amounts of work, so being able to quantify the ratio in the difference between developing features. Other than this, this method proves easy and quite efficient.

Bug Rates:

This is a more obvious metric to measure on as bugs are nearly guaranteed on every project there has ever been. Using this method is fairly straight forward, counting the amount of errors showing up in code and counting re commits to repositories. This method of measurement is reliable and just about globally accessible. Though being nearly an ideal candidate for the set standard, this method has its flaws in if an engineer merely mistypes something, it will come up as a mistake. Yes this is technically a bug but on certain systems it will show up similar to any other major faults, thus making all mistakes equal which doesn't make sense. On top of that counting commits, some developers re commit files when changing spacing or making

comments thus adding to their committing amount. It also isn't a useful way of incentive for employees, as shown in this cartoon.

The summary of the different methods is that there can't be any set globally defined way to measure software engineering, as all projects are different and have specific aspects that are unique to them. These unique attribute means each project needs to prioritise a certain metric more than the others.

Ethics of Analysis

Being able to analyze software engineers and their productivity has been a question of ethics, especially in recent times with the rise of GDPR with massive corporations. Because of these new cyber laws, being able to manage the data at which an engineer uses is difficult and has to be done carefully. Being able to track an employees computer and everything on it is now impossible as with personal PCs retain employees personal information. To efficiently be able to

analyze the data needed, the employee must be aware of all the information being used by employers as to measure their productivity.

Conclusion

To conclude, in this report I have gone over an overview on what can be measured for software engineering, different methods of measuring the productivity of software engineer, and the impact of any ethical issues that could come across when trying to analyze the data.

References

https://medium.com/@yupyork/the-best-developer-performance-metrics-6295ea8d87c0

https://stackify.com/measuring-software-development-productivity/

https://www.infopulse.com/blog/top-10-software-development-metrics-to-measure-productivity/

https://www.7pace.com/blog/how-to-measure-developer-productivity