Measuring Software Engineering

Part I – What Can We Measure?

Measuring the productivity of software developers is a difficult task; one that some would argue is impossible. There are many factors that contribute to what is considered ‘productive’, and measuring only a few will lead to unintended problems.

For example, a basic attempt at measuring productivity would be to count the lines of code a developer contributes in a certain amount of time. This relies on the false assumption that more code equals more work. It is obvious that this method of measuring productivity is highly ineffective, as it just encourages the developer to write unnecessarily long, verbose code in place of concise, understandable code.

The same issue presents itself if you measure how often a developer commits code to a project. This just becomes a measure of how often someone can commit, and not any useful information about what they are committing.

Perhaps we could check how many tasks a developer is completing; or how many bugs they are fixing? But this would just lead to tasks being created for every little thing, and developers intentionally creating bugs just to ‘fix’ them later.

In order to properly measure the productivity of a developer, one must not only be looking at the quantity of work they produce, but also the quality. This is a much

more complicated thing to measure; as there is no easy way for a computer to judge the quality of code.

One option for measuring code quality is to use static code analysis. SCA tools can measure the complexity of code, detect bugs and vulnerabilities, and point out duplication. A developer contributing understandable, bug free code could be considered productive.

Another useful metric for measuring developer productivity is churn. Churn is a measurement of how much of a developer's contributed code is later reworked or replaced. If code is being replaced, then it is likely not a productive contribution to the project.

Measuring how much of a developer's work makes it through a pull request system may also be a useful measurement. If many of a developer's pull requests are being abandoned or closed, they may be considered less productive than a developer who has most of their contributions accepted.

To make the measurement of productivity more difficult, not all productivity can be measured as part of code contributed to a project. A developer could spend a lot of time some days helping other developers with issues. If this is through an internal messaging system, it can also be measured. Messages can be analysed to see if they contain technical language or useful information. However; if they are communicating in person, then it is near impossible to measure. Similarly, if two developers are pair programming, then it may seem to measurement system that one developer is being extremely productive, and the other is doing nothing.

In addition to measuring contribution, physical measurements can also be used to measure and improve the productivity of developers. Measuring a developer's heart rate, GSR, and other factors can give an be used to measure the stress level of developers. Sensors for these measurements could be built into the developer's chair; and the information used to manage the developer's workload to avoid overworking them.

Part II – What Tools Exist?

Many tools now exist to measure the productivity of developers. Most of them are based on a developer's activity in git repositories, be they in Github or Bitbucket.

The first company to create a proper platform for the measurement of developer productivity was GitPrime in 2015. They support all major git platforms. GitPrime gives four key metrics:

Active days – Which days has the developer been committing code.

Commits per day – How often the developer makes commits.

Impact – How much the developer's changes affect the codebase.

Efficiency – The developer's ratio of churned code.

It presents managers with graphs and metrics comparing developers, and can show if developers might be struggling,

Gitprime's most basic package starts at $749 a year, and they have generated about $12 million to date.

In early 2017, GitClear entered the market with their own implementation. Their implementation focuses on having a single metric: lines of code, and the impact of those lines. They have a complex measurement of how impactful a line is; and they base their measures of productivity on that measure.

Their cheapest package costs $295 a month

In mid-2017, Pinpoint entered the market with their own tool. Unlike other tools, Pinpoint does not have the same focus on code analysis as the other tools, and "synthesize data from a range of software lifecycle systems".

Pinpoint provide a long list of factors that are considered in their measurement of productivity. Some are code based, such as the number of commits and rework rate, but most are based on the agile sprint, and how targets are being set and met.

Their basic package is $799 per month, for a team of up to 50 developers.

In 2018, Code Climate entered the market with Velocity. Velocity looks quite similar to GitPrime, and Velocity have published a comparison between Gitprime's features and their own.

Velocity claims that their largest difference is how they focus their metrics on pull requests, as opposed to how GitPrime focus their metrics on people.

Velocity charges $1759 per month for their basic package.

Another tool in the field of productivity measurement is Waydev. Their product seems very similar to GitPrime. They look at the “cognitive load” of the changes that a developer has been making, giving each commit a score based on a number of factors.

Waydev are the only platform to price their product per developer, charging $25 per developer per month.

Part III – What Algorithms Are Used?

Developing an algorithm to measure the productivity of developers is a complex challenge. Each company that provides a solution for this measurement has developed their own algorithm to provide this measurement, each using a different combination of measurements. Most of them are quite opaque about their exact methods, but some, like GitClear, do elaborate on what they measure and how these measurements are factored into their productivity score.

GitClear uses a line count, in combination with a measure of line impact, to evaluate productivity. They account for code churn (is the line later replaced / rewritten), activity type (has the line been added, deleted, moved), domain (is the change to code or a comment), file extension (some languages are more verbose than others, so this is accounted for in this measure), file purpose (is this file for tests, documentation, etc.), duplication (duplicated code is of no impact), related commits (are commits making changes to the same lines), and greenfield implementation (lines changed in large groups are considered less impactful).

Another method, outlined by Jesus Dario in a Hackernoon article (https://hackernoon.com/measure-a-developers-impact-e2e18593ac79) similarly looks only at code contribution. It gives a measure of pull request weight, using the function

W = f (mods, files) + g (adds, dels)

This score is a function of mods (lines added + delated), absolute mods (the number of different lines after the pull request is merged) and changed files.

While not a complete measure (like any lines of code measure, it can likely be easily gamed by a developer who knows how it works), it provides a simple way to get an indication of whether a pull request is impactful; and if a developer if contributing many impactful pull requests, they can be considered productive.

To develop an algorithm to measure developer productivity that is fair is incredibly difficult. It needs to account for the differences between developers (Some developers might make commits more often than their peers, but with less code in each commit), and must be difficult to game, or developers may abuse its flaws to seem more productive than they really are. Considering non-code factors, such as the rate at which tasks are completed, could be a solution to these issues.

Part IV – Is This a Good Idea?

Trying to measure developer productivity brings up an important ethical question: Should we be measuring people?

The very thought of having your work being constantly measured and compared to other developers makes many uncomfortable. It seems like a breach of privacy, having that information measured and stored. And what if a less ethically minded company decided to sell this information to other companies? A metric of how productive a certain developer is could be incredibly useful to employment companies, as companies looking to hire can see how much they can expect to gain from that developer.

We are living in a time where our privacy is rapidly dwindling; is it necessary to gather even more data about people?

Being constantly measured may make more privacy conscious developers uncomfortable, and aside from making people uncomfortable being questionably moral; if developers are unhappy, they will be inherently less productive.

A measure that says X developer is Y productive can effectively create a leaderboard of who is the “best” developer, and, conversely, a “worst” developer. In an environment built around cooperation, introducing a competitive metric of productivity might not be a good idea. While it may drive developers to work harder, and individually be more productive; the productivity of the team relies on cooperation and teamwork, and having developers competing with one another could hamper this teamwork.

Is the bonus productivity gained from utilising these measurements worth the loss of privacy and other drawbacks? In the end, that is for the managers to decide, and then, if it is, to convince the developers that it is so.

Sources:

https://pinpoint.com/

https://codeclimate.com/

https://www.gitclear.com/

https://www.gitprime.com/

https://waydev.co/

https://www.gitclear.com/measuring_code_activity_a_comprehensive_guide_for_the_data_driven

https://hackernoon.com/measure-a-developers-impact-e2e18593ac79

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

https://dev.to/nickhodges/can-developer-productivity-be-measured-1npo

https://nortal.com/blog/the-myth-of-developer-productivity/

https://medium.com/static-object/3-key-metrics-to-measure-developer-productivity-c7cec44f0f67