Philip Healy
18322704

Measuring Software Engineering Report

Introduction

In almost every profession out there, there's at least one metric that a worker can strive towards. If they reach their goal for the day, week, month or quarter, they can reflect and say they've done a good job. They've done what they can and they're satisfied in what they've achieved. A teacher can look at the percentage of students that passed their class. A car salesman can look at how many cars they sold last month. An electrician can look at how many houses they've wired this week. These professions, like many, can be measured in terms of output. There is data that backs up a sentimental feeling that a good day's work has been done. Key Performance Indicators (KPIs) are out there to evaluate the success of a work activity. They are measurable values to demonstrate how effective certain practices are. Software engineering has somewhat lagged behind other industries in terms of data-driven measurement to evaluate performance. In recent years however, analytical platforms and algorithms, using vast amounts of data, have been developed to see if software engineering really can be measured.

Philip Healy
18322704

Can Software Engineering Be Measured?

Software engineering isn't like many other disciplines. It is very difficult to find a metric that shows how well a certain facet of a company is performing on any given day. There has never been a real way for engineers to see progress in their work by looking at data. However, in recent years, managers have been looking for ways to measure the performance of their teams at an in-depth level. There is now technology available to analyse the ways in which workers carry out their projects. How long do they spend on the same task? How often do they take a break? Are they happy from day to day? In software engineering in particular, measurement of commit frequency, code churn and legacy refactoring is used to determine how effective someone's work is. The number of lines of code written should not be a main metric when determining productivity. It is often the case that smaller edits in multiple locations are far more effective than a large edit in one location.

But what about impact? Can one individual's effect on a project really be seen? Can their helpfulness to their team and around the office be given some measurable rating? While you can never put a number on how valuable a person is to a company, the answer to the above questions is increasingly becoming 'yes'. A developer could easily say that they have done three times more commits than anyone else in the office on a given day. This would give the impression that this person has put in a great day's work and his manager should be impressed. More often than not, however, this doesn't tell the full story. The developer may have indeed had large commit volume, but it didn't have much impact if it was the same piece of code being churned out every time. This is why true measurement is needed in software engineering. Managers have the ability to identify where members of their team are getting stuck, identify what meetings are unnecessary and identify fixes for problems. When issues like these are addressed, people have more time to go write quality code and deliver products that customers will love. Instead of relying on gut feeling or subjective clues, managers can look to objective data that give definitive values to measurements. Of course, managers will always have a subjectivity about them in terms of how they analyse their team's performance. Now though, they have the help and guide of metrics to point them in the right direction.

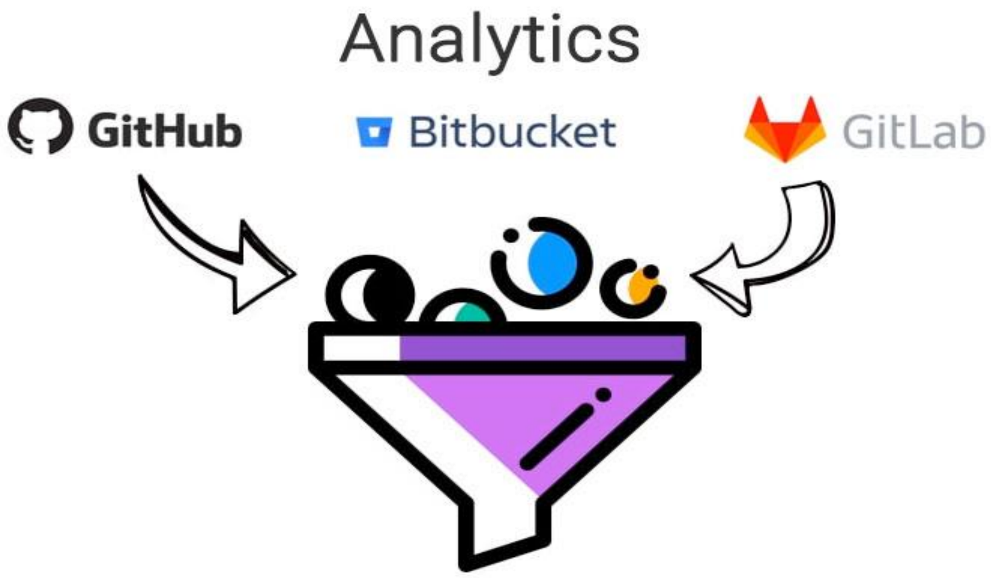
One thing that's important in any industry is a bit of originality. Thinking outside the box. Separating yourself from the crowd. A film I really enjoy is Moneyball. Billy Beane and the Oakland A's go against the typical scouting process and rely on metrics to find the players they need. The emphasis here is on efficiency. If the data is anything to go by, the team is almost guaranteed to find ways to improve. The productivity of the players recruited brings the team to another level. There's nothing better than knowing that the members of your team will be consistent and trustworthy. And these bring results. Like Beane, managers in the engineering world need objective measurements to succeed. They need to find the key indicators that will tell them where improvement is necessary. When they do this, the value of members and their practices should rise. With considered approaches and data to back things up, managers should find to productivity and consistency of their teams at an all-time high.

Happiness among employees is rarely near the top of any manager's list of important factors in the day to day work environment. Why is that? Should happiness not be defined as a key metric? One would think that the happier an engineer is, the more productive they are. If the workplace is made to be a more enjoyable one, surely the quality of output reflects that. There are so many perks to working in some of the bigger firms that must relax employees, keep them satisfied and provide a momentary relief from stress. Looking around a modern office, you can see bean bags on the floors, bright colours everywhere and even free breakfast spreads and snacks throughout the day. The working environment for so many engineers provides the perfect shelter to produce their best work. So, managers must ask the question of whether their employees' happiness can correlate to better results.

When thinking of measuring software engineering, the Personal Software Process is something not to be ignored. It is a way of using tried and tested processes that ensure consistency and quality. In essence, it is a tool used by engineers to guide them from the start to end of whatever project is at hand. It takes measured planning, tracking and goals into account in order for the optimum output to be achieved. Compared to other possible ways of using data to measure performance of teams or groups, the PSP is individualised for every engineer that employs the process. One can analyse past errors, bugs or mistakes and ensure they do not occur again. The Personal Software Process shows that measurement of software engineering is not only possible but, is a basis for improvement in a data-driven world. Without the availability of data and metrics, engineers would not be able to define the most effective process for themselves. Once again, everything is geared towards optimisation of productivity and quality.

Platforms For Measuring Software Engineering

So, if we know that measurement of software engineering is possible, where can we go about finding tools to do such a thing? The choice of answers shows just how prominent data-driven metrics are in the modern world. Git Analytic tools have really propelled the measurable aspect of software processes. They bring an objective, real-time and quantifiable view of productivity and output. They help managers overcome engineering performance challenges like a lack of visibility, poor teamwork, lost project momentum, unnecessary costs and low transparency. At the top of the tree in terms of these tools are Waydev, Pluralsight Flow (formerly known as GitPrime) and Code Climate.



These platforms help managers ensure their teams meet deadlines, reduce wasted time, increase value and boost efficiency in general. Possible roadblocks that are preventing projects from moving forward can be addressed and eliminated with these tools. The methodology behind data-driven engineering is that it all leads towards maximum potential being reached. Essentially, Git Analytic tools reach into vast amounts of code in the cloud and present data for managers to analyse with the ultimate goal of improving productivity. With these tools, managers get to truly know the ins and outs of their team. Benefits of such platforms include eliminating redundant meetings, decreasing time spent on reporting, maximising code time and supporting sharing of knowledge and collaboration. With all these benefits, engineers can feel more satisfied with the work they've done, managers know they've gotten the best output and customers can be satisfied with their resulting products.

A benefit of using Pluralsight Flow is the range of control systems with variable workflows. You can connect to your Git account, whether it be GitHub, Bitbucket etc, to import your repositories. From GitHub or Bitbucket, you can arrange your repositories according to its tag. A contributor can be integrated into teams or concealed from reports. Waydev provides integrations with GitHub, GitLab, Bitbucket, Azure DevOps, and their on-premise counterparts.

The data span of the main Git Analytic tools is much the same. The differences stem from the insights they focus on. Gaining an equal number of sources builds can be met with much complexity. It is important to note that analysis needs to be transparent and insightful. If there are incomplete software builds, there is a lack of precision in the data that is gathered.

The platforms import information from popular Git applications and focus is on the code-level metrics and the code review process. They have a way of carrying out the code review process so it can be quantified into actionable metrics, which can enhance an engineering manager's visibility over their team's productivity. These insights can be used to spot a blocked pull request, incentivise engineers to collaborate more or adjust the workload.

Access to data allows for improved communication between leaders and teams. There is greater transparency between each level of members. There are reports and data to back up gut feelings. Autonomy within teams can be encouraged because platforms are available to allow objective data to speak for performance. When autonomy and trust is present in a team, things like cycle time can be reduced and continuous delivery can be achieved.

How Analytics Are Performed

Quality in software engineering is more important than ever. There are more competitors now and every company fights for every client. Each company wants to improve its service and products. All team members, such as engineers, testers and developers are in control of a product's quality, and they should strive for the best product as is possible. Therefore, they need the most effective algorithms and metrics to find out what processes will give the optimum results.

Contributions in software take on no specific form. Distinguishing them requires more care than just looking at us how many lines of code have been written. For example, it is not simply the case of comparing an engineer who contributes 200 new lines of code to one file with another engineer who looks at three files and adds 40 lines in a few different locations. While there is likely a significant difference in what each engineer offered, neither contribution can be looked at as just a number of lines of code. There are different paths that engineers may go down when writing code. They often have to read old code, invest time in understanding the original code, check whether any changes might damage existing code, carry out the changes, remove any code that is irrelevant and then they have to review everything afterwards.

So, what are the algorithms and metrics used in software and how can they be tracked? A software metric is a measure of software characteristics which are quantifiable. They are important for many reasons, including measuring performance, finding worth, measuring productivity, among others. They are related to four functions of management, which are planning, organisation, control and improvement.

Applying measurement to different areas can help you identify where change is needed. The goal of analysing software metrics is to determine the quality of a current product or process. Once a development project is complete, managers use software metrics to identify, prioritise, track and communicate any issues which leads to better team productivity. The sooner managers can detect software problems, the easier and less expensive the troubleshooting process.

Metrics offer an assessment of the impact of decisions made during software development projects. Software metrics must be simple and computable, consistent and unambiguous (objective), and in consistent units of measurement. It should not matter what programming language is used – the metrics should be independent of this. They should be easy to calibrate and should be adaptable. Furthermore, they should be cost effective to obtain. Metrics linked to goals allow for focus to turn to reducing the amount of code churn, reducing the number of bugs, speeding up task completion and generally improving workload.

In terms of analytics used, there are agile metrics such as lead time, cycle time and velocity, and production metrics such as active days, failure and repair time, productivity and code churn. In addition to these, there are formal code metric such as lines of code and code complexity, as well as customer satisfaction metrics, such as customer satisfaction. That final metric could ultimately be the maker or breaker of many engineering teams.

Impact can answer the question of how much cognitive load an engineer carried when implementing changes, taking into account the amount of code and surface area of the changes i.e. the number of added locations. Other things examined are the number of files affected, how severe changes are to existing code. Impact can often show that certain team members were being overlooked when in fact they were key factors in past successes.

Is This All Ethical?

We've seen that software engineering can definitely be measured. We've found that there are plenty of tools out there that will help managers find out more about their teams. Whether it be impact of commits, time wasted during meetings, or general efficiency, there are so many ways of seeing how software engineers perform, why they perform the way they do, and if these ways are best for the overall output of a team or project.

However, should so much information be available to managers? Should there not be line drawn that indicates that some measurement is too much? Should engineers not have aspects of their work that's free from measurement and possible scrutiny?

Well, when collecting and operating on data that is not publicly available, some discomfort must be introduced to the subjects of the analysis. The scale and ease with which analytics can be conducted changes the ethical framework. There is a remarkable size to the source pool in which personal data can be stored, and hence accessed. More than often with these advanced forms of data collection, employees must put up with intrusive or distracting methods of data collection in order for companies to gain insight into their workflow.

Furthermore, any information and data collected is private. The security of all this information must be guaranteed. The more data that's collected from employees, the bigger the risk that some may be leaked. Safety measures such as firewalls and authentication processes become more and more important as the size and detail of data increases.

Improper access to personal information and the misuse of information could be a block to the measurement of software engineering in the future, should proper precautions not be put in place.

I suppose the benefits most certainly outweigh the risks since so many companies take part in data collection and analysis. However, there must be privacy concerns in play. Issues of complaints about the automated nature of data collection and unhappiness with the invasion of privacy due to insights gained into workflow.

Conclusion

To a degree of limitations, software engineering can certainly be measured. There are undoubtedly measurable qualities that exist in software. Because of the existence of these qualities, platforms are in place to gather and process data, returning unbelievably valuable information to those concerned. Algorithms and metrics are out there to perform analysis on huge amounts of data and information. Remember, this practice is still relatively young and maturing. The degree to which software engineering is currently measured may only be a smidgen of what is possible in years to come.

References

Pluralsight - https://www.pluralsight.com/

'Tips to Choose the Best Engineering Productivity Software' - Waydev, February 7th, 2020. https://waydev.co/gitprime-vs-waydev-vs-code-climate/

Code Climate - https://codeclimate.com/

'Agile Data-Driven: Delivering Better Software Faster' – Waydev, November 19th, 2020. https://waydev.co/agile-data-driven/

'Software Engineering: Why has is eluded data-driven management?' – Pluralsight, 2018. https://www.pluralsight.com/content/dam/pluralsight2/landing-pages/offers/flow/pdf/FLOW_2018-ebook_ar_vf.pdf

Humphrey, Watts S. 'The Personal Software Process(PSP)' – Carnegie Mellon Software Engineering Institute. (November 2000)

'Top 5 Software Metrics to Manage Development Projects Effectively' – Sealights. https://www.sealights.io/software-development-metrics/top-5-software-metrics-to-manage-development-projects-effectively/

'Impact: A Better Way to Measure Codebase Change' – Pluralsight, December 1st, 2016. https://www.pluralsight.com/blog/teams/impact-a-better-way-to-measure-codebase-change