Software Engineering Measurement Report – Tim Kelly

Metrics are important to any business. For engineering teams, these metrics can be used to empower you to measure productivity with the intent of reaching a peak performance and reducing points of friction in the development pipeline. The choice of metrics to observe will decide what behaviour you want to incentivise and can both cause great success and/or lead to potentially undesirable side effects.

In traditional waterfall software projects, it was assumed that software could be specified in advance and quantified by estimates. It was also assumed that the software specification would meet end-user requirements (which often was not the case). This led to an emphasis on in-process measurements like man-months or active days, Lines of Code (LOC), and even number of pages of documentation. In agile development projects, the emphasis is on outcome metrics like story points completed, production defects or customer satisfaction. [1]

Section 1 - How one can measure engineering activity

The methods that can be used for measuring software engineering have evolved to extremely complex metrics. The idea of just commit count, or lines of code is too easily gamed and there is little evidence of proven correlation with business success. There is a variety of different metrics to measure engineering activity and productivity. The more common ones I have found to be used in the industry [2] tend to fit the three criteria.

1. Empirically validated
2. Widely used
3. Gaming-resistant

[1] https://www.sealights.io/software-development-metrics/top-5-software-metrics-to-manage-development-projects-effectively/

[2] https://medium.com/@infopulseglobal_9037/top-10-software-development-metrics-to-measure-productivity-bcc9051c4615

Leading Software Engineering Metrics

OKR-Driven

Objectives and key results (OKR) is a goal setting framework used by individuals, teams, and organizations to define measurable goals and track their outcomes. The development of OKR is generally attributed to Andrew Grove who introduced the approach to Intel during his tenure there. [3] OKRs are used by companies such as Spotify, Twitter, Samsung, Microsoft, and Google. [4] One of there greatest benefits is that they are hand crafted to optimise business value. Although, this can be time-consuming, and the Key Result can lie beyond the control of the developers. Another benefit is that, when done right, OKRs are hard to game as they are chosen in accord with business objectives. Poor Key Results such as "introduce 10% less bugs this cycle" could be gamed just by not introducing any new innovations.

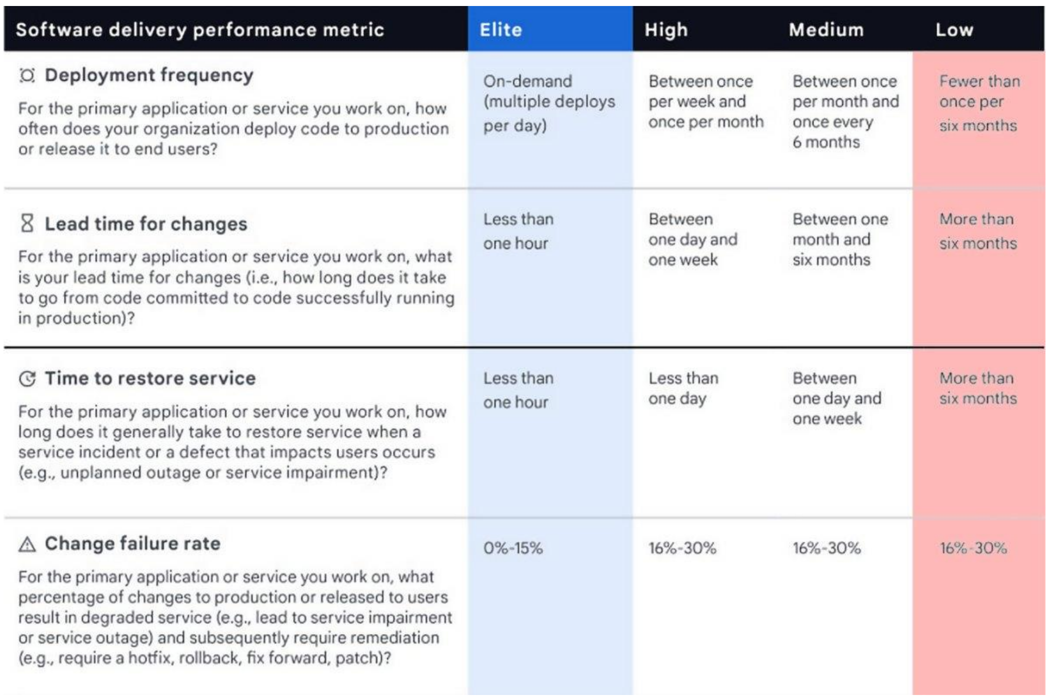
Story Points Completed

Story point estimation is a technique used in Agile project management to replace task estimation in time or money. [5] There are a host of different ways to calculate story points, it comes down to a combination of time and effort to complete some task. A good management can use this method to effectively as it is project specific and the longer you work with the team, the more accurate you can estimate the value of the story points.

Google DORA DevOps Metrics

Google Cloud's DevOps Research and Assessment (DORA) team has conducted a seven-year research program. This research has validated a number of technical, process, measurement, and cultural capabilities that drive higher software delivery and organizational performance. [6]. Each of Google's four recommended metrics offer a scale from "Low" to "Elite", as seen in fig 1.1, to rate your teams' efforts.

[3] https://corvisio.com/what-is-okr/
[4] https://www.tability.io/odt-articles/okrs-scrum-and-evidence-based-management
[5] https://teamhood.com/agile/story-point-estimation-table/
[6] https://cloud.google.com/devops



Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Fig 1.1: Shows the four criteria recommended by Google DORA and a scale to rate performance of a development team.

Deploy Frequency. Googles definition for Deploy Frequency is

> "how often does your organization deploy code to production or release it to end users?" [7]

This is the most gameable of the metrics they suggest. Probably the most useful insight that could be gained from this metric is the fluctuation over time.

Lead time. Googles definition for Lead Time is

> "what is your lead time for changes (that is, how long does it take to go from code committed to code successfully running in production)?" [8]

With the "Elite" grade being granted to those teams that can achieve this in under an hour, it is obvious that this favours an automatic process that bypasses manual quality assurance signoffs. With continuous integration and continuous deployment process becoming more popular in

[7] https://www.devops-research.com/quickcheck.html
[8] https://www.devops-research.com/quickcheck.html

software engineering companies in the past decade, this makes sense why this metric should be considered.

Time to Restore Service. Google defines this as

> "how long does it generally take to restore service when a service incident or a defect that impacts users occurs (for example, unplanned outage, service impairment)?" [[9]]

There is a massive variation in this metric between "Elite" (one hour) and "Low" (more than six months). Gathering the data for this research would yield a significant amount of variance as well just by the nature of different companies having different environments for update reviews.

Change Failure Rate. Googles definition for this is

> "What percentage of changes to production or releases to users result in degraded service (for example, lead to service impairment or service outage) and subsequently require remediation (for example, require a hotfix, rollback, fix forward, patch)?"[10]

This is the measurement of critical bugs to release. According to Google, this is the least likely of it's metrics to be gamed.

There is no agreed upon best metric, but the most unison I see with the idea that it can't be measured. This is interesting because if so many people agree that it can't be measured accurately, or that it never will be measured accurately, why are there so many tools created to measure what are agreed upon as flawed metrics? An argument can be made that these tools only provide quantitative data, and it is up to organisations how they want to interpret these results. I think it can be considered irresponsible to provide data in the way some of these applications are doing so as it can lead to unfair grading of engineers.

[9] https://www.devops-research.com/quickcheck.html
[10] https://www.devops-research.com/quickcheck.html

Section 2 – Platforms on which one can gather and perform calculations over these data sets

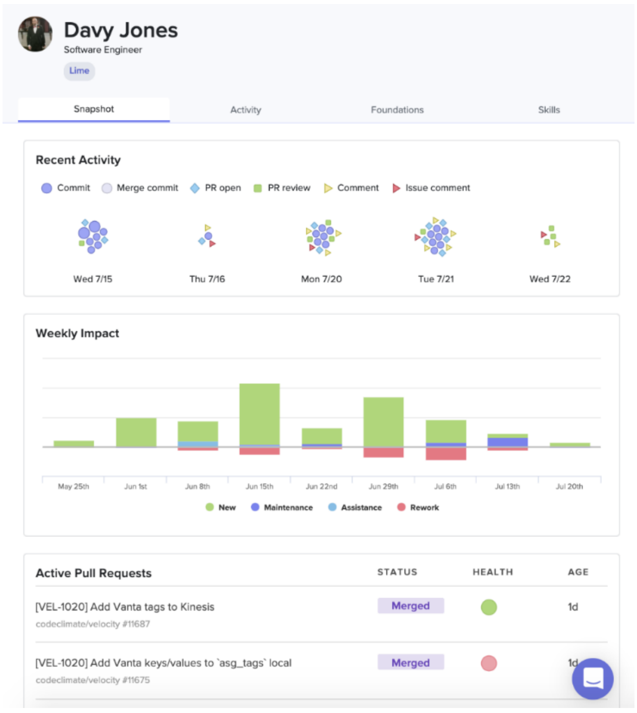
There is a plethora of applications out there to measure software engineer activity. These applications are created so that someone can only vaguely understand how an engineer does their job but allow them to inn some ways appropriately grade their work. Let's start with two applications that take a broad view over how they measure engineering and do so in different ways.

Velocity and Plurasight Flow (previously GitPrime).

What is velocity? According to their launch, "Developer360 gives you instant visibility into your developer's active work, improvements along key metrics, and skills." [11] The report contains four sections:

1. **Snapshot**

   This shows what work the engineer is currently doing and what impact it has. This is intended to be utilised by managers to prevent bottlenecks down the line. See fig 2.1.

Active Pull Requests	STATUS	HEALTH	AGE
[VEL-1020] Add Vanta tags to Kinesis codeclimate/velocity #11687	Merged	●	1d
[VEL-1020] Add Vanta keys/values to `asg_tags` local codeclimate/velocity #11675	Merged	●	1d

   Fig 2.1: The snapshot section of the Velocity Developer365 Snapshot dahsboard.

[11] https://codeclimate.com/blog/developer360-launch/

2. **Activity**

   This is a dashboard presenting the past month of work of the engineer. This is intended sop managers can see and manage the workload of the engineer.

3. **Foundations**

   To show how a member of the team is trending according to the Velocity metrics.

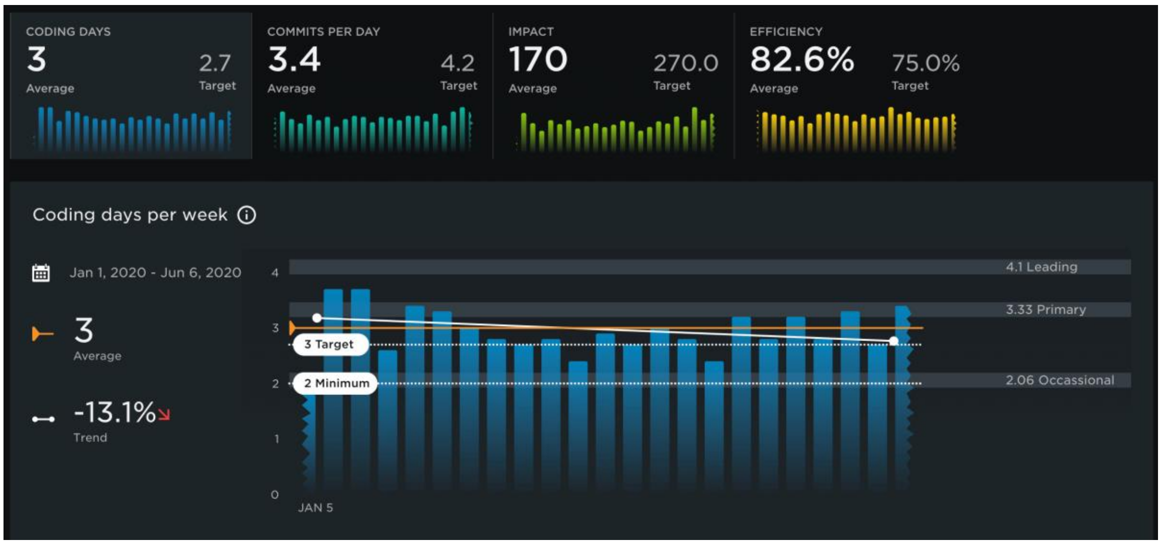
4. **Skills**

   Displays the languages that a developer has been working with.

The Velocity Developer365 report is do provide objective metrics and not to presume what they may indicate.

And Plurasight Flow, “Accelerate velocity and release products faster with visibility into your engineering workflow. Flow aggregates historical git data into easy-to-understand insights and reports to help make your engineer teams more successful.”[12]

Plurasight Flow provides a more contributor-based report. It’s dashboard displays data gathered from the version control system used, such as Github, and displays it in a comprehensive dashboard. See fig 2.2.



[12] https://www.pluralsight.com/product/flow

Fig2.2: This shows various statistics measured of the engineer, based on the commits provided. It also compares this data to target values, as well as some top performers in the team.

DORA metrics

Since Google released these metrics, applications such as Blueoptima [13] allow you to track these metrics by integrating their software with your version control system.

Software not specific to software engineering

Using metrics like OKRs, there are many applications not specific to software engineering that can be used. Weekdone [14] is an application for companies to help track OKR objectives, as well help organise teams with smaller goals and weekly plans. This is a less automatic method of tracking progress but can still be useful. See fig 2.3.

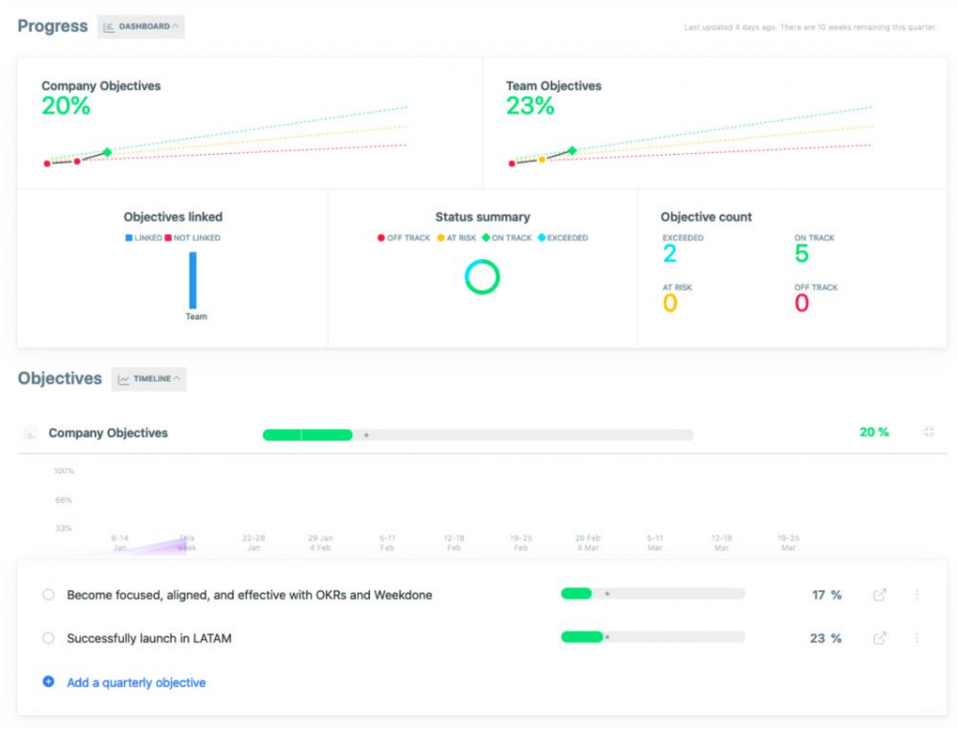


Fig 2.3: A screenshot of the Weekdone dashboard, showing how it displays progress towards achieving the Key Results.

---

13 https://www.blueoptima.com/
14 https://weekdone.com/

Section 3 - Various kinds of computation that could be done over software engineering data, in order to profile the performance of software engineers

There is no definitive right answer to measuring an engineer's productivity, and many agree that is a question that will never be completely solved. That being said, many companies do gather this data, and there are several ways in place that people are trying to profile the performance of their engineers.

A simple method of this could be implementing the Halstead Metrics. By counting the tokens and determining which are operators and which are operands, the following base measures can be collected:

n1 = Number of distinct operators.

n2 = Number of distinct operands.

N1 = Total number of occurrences of operators.

N2 = Total number of occurrences of operands.

n1* = Number of potential operators.

n2* = Number of potential operands.

[[15]]

The Halstead method uses these to identify program length, volume, and difficulty and can be used over datasets to evaluate an engineer's work. This is a very basic but is simple to calculate, can measure overall quality of code and can predict error rate. There are drawbacks though. It depends on the complete code, and it has no use as a predictive estimating model.

Artificial Intelligence or Machine Learning techniques are being used throughout the industry to efficiently analyse large amounts of data and are particularly useful for situations like

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

analysing programmer productivity. This is because AI technologies such as ML can be used to analyse big data and provide more significant insights that can improve decision making regarding automated infrastructure management, environment setup and code deployment. [16] This can inform management of trends in employee activity and help identify ways to increase productivity.

An organisation could gather data on its engineers and try to identify factors that contribute to high return on investment. It could then attempt to identify these traits pre-emptively in order to maybe divert assets to increase return on investment, or to identify top performers that they might want to incentivise.

It could do this through methods such as unsupervised learning, where it provides the system with data sets containing data points that are neither classified nor labelled. The algorithms then are to discover correlating factors of their own and classify, group, and/or label the data. Once the system is trained, it should then be able to identify key factors that lead to productive engineers. This system could have the adverse effect of promoting legacy biases. An example of this is Amazon attempted to make an AI in 2014 to automate their hiring process, but as there is historically a high disparity between male and female software developers, the AI soon started to highly favour men over woman. Amazon eventually scrapped the project. [17]

A way around this could be to implement reinforced learning. This is similar to unsupervised learning, but the output is provided with feedback. This could be useful as it could help organisations with preventing the rewarding of unwanted behaviour like gaming as a reinforced learning system will learn to identify this as negative behaviour.

[16] https://wiprodigital.com/2019/09/03/future-of-software-development-ai-in-developer-productivity-part-2/
[17] https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G

Section 4 – Ethics

The use of gathering data off these metrics in the assessment of engineering has ethical concerns. What kind of data should be allowed? Does the engineer get to make clear his own boundaries? How transparent does the company need to be when gathering this data?

Data Collection

An obvious ethical concern to consider when collecting data is where to draw the line. The more you know about an employee, the better you can both predict and assess the work done by that engineer. Knowing an employee's health could help an employer with understanding their work. This could help prevent burnout and would lead to better work in the future. But the employee should have the right to limit what his employer collects if it is too personal. According to research done by the Kaiser Family Foundation through annual surveys, 14% of employers who offered health insurance in 2017 collected data from their employees' wearable devices, and that number jumped up to around 20% in 2018. [18] If we accompany this with the research done by the Health Enhancement Research Organization, Brigham Young University and the Center for Health Research at Healthways shows, employees with healthy eating habits outperformed their colleagues with lesser eating habits and had 27% lower absenteeism. Also, workers who exercised three times a week were 15% percent more likely to have better job performance. [19] This makes a very clear case for why a company might want to track an engineer's health, but an employee does not belong to a company and in my opinion, should be able to live the lifestyle he chooses.

Data Storage

The collection of this data brings upon the obvious risk that this data can be breached. A malicious attack on Yahoo in August 2013 by a group of hackers meant that 3 billion accounts were compromised. [20] The data stolen security questions and answers, increasing the risk of identity theft. Not all data breaches are malicious. The City of Calgary in Canada is being sued for $92.9 million for a 2017 privacy breach impacting more than 3,700 of its employees. The

18 https://www.advisory.com/en/daily-briefing/2019/02/20/employee-wearables
19 https://news.byu.edu/news/poor-employee-health-means-slacking-job-business-losses
20 https://www.upguard.com/blog/biggest-data-breaches

accusation being that the city was "acting with the most obvious neglect" because an employee of the city sent an email to an employee in another Alberta municipality sharing Workers' Compensation Board claim details, medical records, Social Insurance Numbers, addresses, dates of birth, Alberta Health Care numbers and income details. [21] With data being gathered, you have to consider what this data being leaked would mean. A company should consider this when deciding what data, they should gather on an employee. Just to look at one example of many, if we look back on the idea of collecting data on employee health, a leak of this data could be used against an employee by a health insurance provider as it might suggest a lifestyle, such as a sedentary one, that could make them more likely to suffer of some medical conditions, such as diabetes. [22] The General Data Protection Regulation (GDPR) attempts to recognise some of these concerns by adding accountability and some restrictions on the data gatherers end, but these rules are simply a step in the right direction, not a solution to all the problems that may occur.

Unfair metrics

The collection of this data comes with a responsibility to use appropriately. Many of the ways of measuring an employee are done with AI and machine learning techniques. An AI can simply only take the data given to it and this can be easily misled. If we use an OKR driven method for measuring productivity and the engineer has been given a poorly conceived goal my management, the AI sees an employee that didn’t achieve its goals. This is a very simplistic idea of an issue, but artificial intelligence tools are built by people and by using the same metrics, we can end up creating just an automated system with all our faults. An article by foley shows several times large companies attempted to us AI for hiring but ended up with software with racist or sexist bias. [23]

Misuse of Data

A lot of my feelings on this issue come down to not being overly reliant on just these metrics. Good work speaks for itself. Insights on this data could be good for informing companies on

[21] https://www.redteamsecure.com/blog/danger-ranks-7-times-employees-caused-data-breaches
[22] https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight
[23] https://www.foley.com/en/insights/publications/2019/02/is-artificial-intelligence-sexist-and-racist

large decisions but, as we saw above, every metric has its flaws. Googles DORA methods list deployment frequency as a metric. A high stress environment might lead an engineer to be poor pedantic about their work and spend more time making sure code is high quality before deploying. Understanding the situation of each engineer should be an important part of measuring overall productivity.

Conclusion

I think that the information that companies should be allowed to gather on their employees and the data analysis done to measure an individual’s productivity should be kept to a minimum. Monitoring a team’s performance and trying to improve return on investment is necessary when running a business but when you attempt to automate this process you are affecting a human. An engineer should be able to work at their own rate and people go about problem solving in different ways. When you attempt to create metrics and monitor trends, it can penalise outliers. Even if they are good employees, just in non-traditional ways.