Using Big Data and Analytics to Measure Software Engineering Activity Report 2019

Author: Laura Brennan (Student ID: 17331720)

Summary

The purpose of this report is to consider the issue of measuring workers in order to assess and enhance performance, and in some cases augment or replace with automated systems. In this report I will consider the ways in which the software engineering process can be measured and assessed in terms of the following four categories:

1. Measurable data
2. The computational platforms available to perform this work
3. The algorithmic processes available
4. The ethical concerns surrounding these kinds of analytics.

1. Measurable Data

This section looks at what data is being measured to gain insight on the productivity of software engineering activity. It discusses common measures of software engineer's productivity, comparing advantages and disadvantages of each, the personal software process model to track performance, and finally a glance to future possibilities of data sources.

`Software metrics' is the term used to describe the wide range of activities concerned with measurement in software engineering. The metrics main purpose is to provide information to support managerial decision-making during software development and testing (Fenton & Martin, 1999).

What are companies measuring?

When it comes to evaluating your team of software engineers, the metrics used will play an extremely important role. For this section I will look at what data, in the context of software engineering, is measurable and being measured to potentially tell us something about how software engineers are working.

Time Driven Measures:

Velocity

- This is the average amount of work a scrum team completes during an iteration (known in agile as a "sprint").
- In agile methodology this is measured in either story points or hours.
- Story Points:

  - Each story point represents a normal distribution of time.
  - For example, 1 story point could represent 4-12 hours, then 2 could represent 10-20 hours. They are an abstract measure of effort, indicating to the team the level of difficulty of a task or "story". Difficulty could be related to risks, complexities and efforts involved.
- Used as agile process metric.

Defect Removal Efficiency

- A measurement of how efficient your team is at removing bugs from the product before they affect the customer.
- Take the total number of bugs that you find in development and you divide it by all bugs (those found in development AND those found in production) and multiply the result by 100 to get your Defect Removal Efficiency rate, as a percentage.

Lines of Code written in an hour

- This numerical metric looks at how many lines of code an employee writes per hour.

Quality/Effort/Cost Driven Measures

Peer code reviews

- A way of measuring developers code quality whereby developers review each other's codes to see if they reveal minimal problems and discrepancies.
- This will also bring to attention the developer's level of adherence to coding guidelines and techniques.

Number of Bugs/ Defect Density

- Defect Density is calculated as the number of bugs found during the test period divided by the thousands of lines of code for that application, module, or service.
- As an example, if you've found 20 bugs in your service and 2 thousand lines of code were changed, then your Defect Density is 10 (Medium, 2017).

Mean Time Between Failures and Mean Time to Recover/Repair

- Both measure the software system's performance in its current production environment.
- Mean time between failures is the predicted elapsed time between inherent failures of a mechanical or electronic system, during normal system operation. MTBF can be calculated as the arithmetic mean time between failures of a system.
- Mean time to repair is a basic measure of the maintainability of repairable items. It represents the average time required to repair a failed component or device.
- Used as agile process metric.

Interaction with Training

- Measures employee's engagement with training videos, computer- based training and any other trackable training software.
- Can include employee's involvement in training other employees on their weaker areas.

Contribution to the Code Base

- A measurement of efficiency can be measured as the percentage of an engineer's contributed code that is productive (Ambition & Balance, 2019).
- The higher the efficiency rate, the longer that code is providing business value.

Code Churn

- The percentage of a developer's code representing an edit to their recent work.
- Typically measured as lines of code (LOC) that were modified, added and deleted over a short period such as a few weeks (Ambition & Balance, 2019).
- The most prolific engineers contribute lots of small commits, with a modest churn rate, resulting in a high-efficiency rate.

Code Coverage

- A measurement of how many lines/blocks/arcs of your code are executed while the automated tests are running.
- Code coverage is collected by using a specialized tool to instrument the binaries to add tracing calls and run a full set of automated tests against the instrumented product.
- Makes sure they accounted for base cases and edge cases.

Advantages and Disadvantages of Different Metrics for Measuring Software Engineering Productivity:

Metric	Advantages	Disadvantages
Velocity	• Very useful for forecasting (story points and hours) (Medium, 2017).	• Should only be used to estimate and plan iterations, not useful for comparing. • Based on non-objective estimates (Lowe, 2019).
Defect Removal Efficiency	• High DRE rate might indicate a high Quality of released code. • Gives you an idea of how good your software engineers are at getting out ahead of issues that might affect customers. • Useful for planning for resources, assessing your pull request process, and	• Must balance this metric with an eye on Speed and Accuracy. • Does not measure software engineer's efficiency at removing bugs when client is currently using it and system fails.

	keeping track of the trending health of your product.	
Lines of Code written in an hour	• Give a numerical metric to measure employees output (Medium, 2017).	• Measures quantity over quality • Does not account for hours spent problem solving, thinking through solutions and tests, where developers are not producing any code.
Peer Code Reviews	• Easy to implement • Shown to be effective in finding bugs and knowledge-transfer • Some developers like it (Stackify, 2017)	• Some developers do not like it • Time-consuming • Usually no verification that defects are actually fixed (Stackify, 2017)
Number of Bugs	• Allows you to identify high priority bugs	• Can foster blame culture • Can lead to arguments about whether bugs are caused by development or by poor quality analysis • Can cause defensive programming, which will increase the maintenance costs. Focusing on bugs leads to more bugs. • Can slow down developers as no code means no bugs.
Mean Time Between Failures/To Repair	• Allows you to see if developers are becoming more effective in understanding security issues such as bugs and how to fix them. • Quantifies how well software recovers and preserves data	• Can be difficult to link to individual employees rather than teams.
Interaction with Training	• Shows employees invested in growing their own skillset or helping others. • A "positive and virtuous" practise that could encourage friendship between colleagues	• Can be complex to track engagement in mentoring other employees.

	(Ambition & Balance, 2019).	
Contribution to Code Base	• Quantifies impact on the code base. • Measures how effectively employee individual programmer is performing (Stackify, 2017).	• Complex to measure impact and implement correctly in some cases (Stackify, 2017).
Code Churn	• Allow software managers and other project stakeholders to control the software development process, especially its quality. • Understanding an engineer's typical efficiency rate can help you understand their character and where they will fit in best.	• Does not provide context, reasons can vary for example: ○ Indecisive product team running a developer in circles ○ Developers under-engaged ○ Developers experiencing difficulty with a feature (Kimmel, 2019).
Code Coverage	• Good at measuring how many code paths are hit by a test. • Identifies areas of code that are not tested.	• Can give false positives as does not mean the behaviour of the code is correct. • Does not tell you if tests are comprehensive or on target.

Personal Software Process

- Personal Software Process (PSP) is an interesting way of looking at software engineer's productivity.

- It is a structured software development process designed to help software engineers better understand and improve their performance by bringing discipline to the way they develop software and tracking their predicted and actual development of the code (Bjorn, 2018).

- Shows software developers how to plan and track their projects, use a measured and defined process, establish goals, and track their performance against these goals.

- Assists engineers in managing software quality from the start of a project to completion, analysing the results of each task and using the results to improve the software process of the next project.

A Glance to the Future – What Data *Could* We Gather to Measure Software Engineers Productivity?

As more data is available to us through self-monitoring technologies such as the Fitbit, we will be able to use more data to supply metrics, such as:

- Tracking locations
- Tracking groups
- Tracking tone of voice
- Tracking peer programming
- Tracking length and frequency of coffee breaks
- Tracking length and frequency of bathroom breaks
- Heart rates after committing to codebase
- Heart rates interacting with certain employees
- Posture
- Frequency and nature of employee interactions

The influence model has been applied to various social systems, particularly those that have been monitored by sociometric badges. These badges are personal devices that collect individual behavioural data such as those listed above, including audio, location, and movement. Early attempts to analyse data from these badges focused on questions revolving around group interaction and interpersonal influence (Pan et al, 2012).

If we find that behaviour between two individuals is correlated, it could be due to influence, but it could also be due to other factors such as selection (I choose to interact with people like me) or to contextual factors (you and I are both influenced by an event or a third party not in the data). This does not make observational data worthless. It just means that we should carefully consider alternative mechanisms that may underlie correlated behaviour (Pan et al, 2012).

2. Computational Platforms Available to Perform this Work

More companies are emerging suggesting they can provide analytics capabilities. Big data and analytics have come to be at the forefront of corporate agendas. As data-driven strategies take hold, they will become an increasingly important point of competitive differentiation (Barton and Court, 2013).

This section of the report explores how we can infer information from raw, historical data by looking at computational platforms available, as well as newer emerging data sources and exploring what data we could gather that is not currently used.

Extracting Value from Raw Data:

Computational platforms allow us to take large volumes of data and build our own analytics, extracting value from the historical data gathered from measuring software engineers.

Modern operations-monitoring software makes gathering detailed metrics on individual programs and transactions quite easy, as programmers are building code sitting on top of analytics platforms. However, we must be aware that it takes time and careful thought to set up proper ranges for alerts and/or triggers for scaling up or down, for cloud-based systems (Lowe, 2019).

An example of these application service products:

GitPrime:

- Application service product allowing you to push data analytics.
- Aggregates historical git data into easy to understand insights and reports, to help make engineering teams more successful.

R Programming:

- R is the leading analytics tool in the industry and widely used for statistics and data modelling.
- Earlier mining software depository allowing you to do deep analytics over code.
- R is an open-source project supported by the community developing it.
- However, some companies strive to provide commercial support and/or extensions for their customers (En.wikipedia.org, 2019).

New Emerging Data Sources:

The "Quantified Self" movement has tremendous potential in the workplace with new technologies capable of monitoring employee's location, heartbeat and much more.

An example of these self-monitoring products:

Fitbits:

- Fitbits are growing in popularity for workplace analytics.
- They have several metrics available to monitor individuals, including:
    - GPS
    - 3- axis accelerometers
    - Digital compass
    - Optical heart rate monitor
    - Altimeter
    - Ambient light sensor
    - Vibration monitor

3. Algorithmic Approaches Available

This section will look at the commodity infrastructure of the algorithmic approaches available. It examines what kind of processing is being done with these data insights, how are people using these algorithms in relation to decision making and if scalability is being prioritised above accuracy and usefulness of these algorithms.

What Kind of Algorithms are Companies Using?

We have established the kinds of data that can be gathered, but now must look at ways these data insights are being utilised by companies.

Higher Order-Data:
(Note: Metrics used will depend on definition of question for example what the company deems as “value”, “engaged”, what a certain job promotion would require will determine what metrics to base off.)

- What **social groups** are there in the office?
  Could be tracked via employee location, tone of voice and frequency of interactions with other employees.

- What employee is a **flight risk?**
  Could be indicated by level of interaction with training and other employees, engagement in meetings, impact on code base.

- Which employees are **most engaged?**
  Could be indicated through tone of voice, impact on code base, velocity, peer mentoring and engagement with training to grow skill set.

- Who is due a job **promotion?**
  Could be determined through trends in velocity, efficiency, quality of code metrics seen above, communication skills with peers shown via engagement with peer reviews and mentoring.

- Who adds **value** to the company?
  Could be seen via impact on the code base, involvement in training and encouraging other employees, and several different metrics depending on what kind of value an employee adds.

Examples of Companies Using Algorithms (Enthoven, 2019):

- **Fuel Consumption Reduction -** UPS tracked van drivers location using Fitbit, analysed the routes and reduced its fuel consumption by more than a million gallons a year by optimizing driver routes with its "right turn only" rule.
- **Walking Speed and Employee Location** - European supermarket chain Tesco has employees wear electronic armbands so managers can track how fast they walk around the warehouse, but workers complain their bathroom breaks are measured too.
- **Non-Work-Related Activities -**Other companies are eating, exercising, and sleeping habits.

- **Identifying Security Risk Employees -** The federal government has built a system that monitors the work-related and personal activities of more than 5 million of its employees to flag those who might be a security risk.
- **Microchip Implanting Replacing Keys -** Some companies began evaluating microchipping technology as a replacement for issuing keys. The state of California even passed a law making it illegal for companies to implant microchips as this was becoming more of a possibility for firms in California.
- **Microsoft Top 10-** Since the 2000s, Microsoft used a stack ranking system similar to the vitality curve. The stack ranking system was relatively secretive for a long time at Microsoft; non-manager employees were supposed to pretend they did not know about it. They claimed it was to "make sure all employees see a clear, simple, and predictable link between their performance, their rating, and their compensation". The model had 5 buckets, each of a predefined size (20%, 20%, 40%, 13%, and 7%), which management used to rank their reports. All compensation adjustments were predefined based on the bucket, and employees in the bottom bucket were ineligible to change positions since they would have the understanding that they might soon be let go (En.wikipedia.org, 2019).

Automated Decision Making

The above stories raise questions about how far employers can go to keep tabs on their workforce, explored further in the next section regarding ethics. These algorithms bring risks, especially when it comes to decisions usually made by humans starting to be made by machines based on data insights and algorithms.

Metrics used are highly important if companies are using them in algorithms to decide if employees are working hard enough. If the algorithm, for example the Microsoft ranking system, does not deem an employee to be working hard enough, it could automatically rank them poorly and fire them without taking other factors into account like a human would. Managers might know you are going through a tough time due to personal factors and could offer leeway, metrics cannot tell you the whole story, only the team can (Lowe, 2019).

Scalable vs Accurate/Useful Dilemma

We must be aware that it takes time and careful thought to set up proper ranges for alerts and/or triggers for scaling algorithms from metrics up or down, for cloud-based systems (Lowe, 2019).

Monitoring individual behavioural data can be complex, for example, for the inference model there is the prevailing question about these inferences related to their validity: how do we know that the measure of influence is real (Pan et al 2012)? Organisations need to be aware of the possible trade-off in validity in return for scalable algorithms that do not produce accurate, insightful data with useful meaning.

4.Ethics Concerns Surrounding this Kind of Analytics and Recommendations

Data as an Indicator, Not a Decider

When any or all these metrics are out of the expected range or trending in alarming directions, do not assume a cause. Talk to the team, get the whole story, and let the team give the data context before deciding whether there is cause for concern, and if so, how to fix it (Lowe, 2019). You cannot know or assume root causes from the numbers, but these metrics give insight into where your essential processes need attention.

Personally, I think these metrics are useful for highlighting areas of concern for managers but should stop before the point of decision-making.

Context is King

I strongly believe guidelines should be put in place, so we do not lose crucial human interactions that give context to situations, such as in the following examples.

A high velocity rate and low defect removal efficiency across a few iterations, for example, may mean that the production issues are currently a lower priority than new features, or perhaps that the team is focused on reducing technical debt to fix entire classes of issues, or that the only person who knew how to fix them quit, or something else. You cannot know or assume root causes from numbers alone (Lowe, 2019).

A high code churn rate does not automatically mean inefficient developers. An indecisive client may be running developers in circles, developers may be unengaged with project and need management to increase motivation or incentives, or developers may be stuck on a one feature. It may only take a simple conversation with a manager to provide some extra support and fix the problem (Kimmel, 2019).

Privacy- What Data *should* we gather?

There is a delicate balance between protecting your bottom line and your team's privacy. Numerous studies indicate that trust is a substantial factor in employee productivity (Mathews, 2018). While employees knowing you're watching them may seem like it would increase their productivity and waste less time, your team won't feel trusted when you've got eyes on them 24/7 (Mathews, 2018). As Stephen M. R. Covey, author of The Speed of Trust writes, "Trust is not a soft, social virtue — it's truly a hard, economic driver for every organization." According to Fortune Magazine's 100 Best Companies to Work For research, "trust between managers and employees is the primary defining characteristic of the very best workplaces." I think it is unethical to monitor employee's breathing, heart rate, posture, tone of voice and other mannerisms as metrics for productivity as this is not directly monitoring their work but them as humans. I believe guidelines must be put in place to draw the line companies should not cross past, to maintain a level of dignity and privacy for employees.

Beware of Linking Pay to Measured Performance

There is a growing body of evidence that suggests that using performance as a way of measuring pay is destructive (Ambition & Balance, 2019). From my research I would believe it would encourage heroism, increase workplace stress and a blame culture rather than increase efficiency and teamwork, particularly when they are measuring unwanted items. Employees who want to work for you will produce high-quality work in a timely fashion. If employees don't want to work for you, no amount of productivity measurement and optimization will solve that core issue (Ambition & Balance, 2019).

Empower Employees and Encourage Autonomy

As more and more data become available, we should focus less about standardizing human behaviour to measure output and efficiency, and more about empowering individuals and your team. If your software engineering team experiences productivity issues, your first instinct might be to try to take a deep dive into individual performance via quantitative metrics (Ambition & Balance, 2019).

For example, this developer is only creating X lines of code a day. Half the team is only logging four work hours a day on our time-tracking app.

However, your team's desire to work more efficiently comes from factors like whether you demonstrate "positive and virtuous" leadership practices and whether they feel like they're doing valued work that matches their skill-set (Ambition & Balance, 2019)– not because you required them to start logging their time and various metrics.

I think for long term success companies should focus on employee's intrinsic motivation and be wary of the effect these metrics have on it. Promote the idea that increased efficiency is for "mutual benefit." Show employees that producing higher-quality work more quickly is a win-win. Employees work fewer hours, the company grows and invests back in the employees and workplace, making employees more satisfied and more motivated to continue producing at the same level, and that cycle spirals up and up (Ambition & Balance, 2019).

Conclusion

No one has discovered a silver-bullet metric showing managers and company leaders how to measure the true productivity of software engineers. Nevertheless, successful companies have found innovative ways to measure and improve productivity. It may take a different approach and careful definition of what "productivity" is to your team, to be mindful of metrics used especially when important decisions rest on the analysis of those metrics. Instead of trying to encourage productivity through increased metrics which can cross the line to give a "Big Brother" impression to employees, companies could take the approach of making higher-quality work seem like a win-win and granting employees autonomy, so they are intrinsically motivated. Not all metrics are flawed, however I believe when using these metrics and algorithms to track productivity managers must make sure to focus on supporting your team, instead of judging them for perceived lack of productivity, especially before getting to the root cause with the team themselves.

References:

1. Stackify. (2017). Software Development Productivity: Most Important Metrics to Track. [online] Available at: https://stackify.com/measuring-software-development-productivity/ [Accessed 24 Oct. 2019].
2. Lowe, S. (2019). 9 metrics that can make a difference to today's software development teams. [online] TechBeacon. Available at: https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams [Accessed 24 Oct. 2019].
3. Fenton, N. E., and Martin, N. (1999) "Software metrics: successes, failures and new directions." Journal of Systems and Software 47.2 pp. 149-157.
4. Medium. (2017). 13 Essential Software Development Metrics to Ensure Quality. [online] Available at: https://blog.usenotion.com/13-essential-software-development-metrics-to-ensure-quality-219cfc264ed1 [Accessed 25 Oct. 2019].
5. W. Pan, W. Dong, M. Cebrian, T. Kim, J. H. Fowler and A. S. Pentland, "Modeling Dynamical Influence in Human Interaction: Using data to make better inferences about influence within social systems," in IEEE Signal Processing Magazine, vol. 29, no. 2, pp. 77-86, March 2012.
6. Enthoven, D. (2019). How to Track Your Employees' Productivity Without Becoming Big Brother. [online] Inc.com. Available at: https://www.inc.com/daniel-enthoven/how-to-track-your-employees-productivity-without-becoming-big-brother.html [Accessed 3 Nov. 2019].
7. Ambition & Balance. (2019). The Do's and Don'ts of Measuring Employee Productivity in the Knowledge Economy - Ambition & Balance. [online] Available at: https://doist.com/blog/measure-improve-employee-productivity/ [Accessed 27 Oct. 2019].
8. Kimmel, T. (2019). 5 Developer Metrics Every Software Manager Should Care About | GitPrime Blog. [online] GitPrime. Available at: https://blog.gitprime.com/5-developer-metrics-every-software-manager-should-care-about/ [Accessed 30 Oct. 2019].
9. Barton, D. and Court, D. (2013). Three keys to building a data-driven strategy. [online] McKinsey & Company. Available at: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/three-keys-to-building-a-data-driven-strategy [Accessed 30 Oct. 2019].
10. Enthoven, D. (2019). How to Track Your Employees' Productivity Without Becoming Big Brother. [online] Inc.com. Available at: https://www.inc.com/daniel-enthoven/how-to-track-your-employees-productivity-without-becoming-big-brother.html [Accessed 3 Nov. 2019].
11. En.wikipedia.org. (2019). R (programming language). [online] Available at: https://en.wikipedia.org/wiki/R_(programming_language) [Accessed 3 Nov. 2019]
12. En.wikipedia.org. (2019). Vitality curve. [online] Available at: https://en.wikipedia.org/wiki/Vitality_curve [Accessed 4 Nov. 2019].
13. Mathews, K. (2018). Privacy vs. Productivity: How employee monitoring software backfires. [online] RescueTime Blog. Available at: https://blog.rescuetime.com/employee-monitoring-privacy-vs-productivity/ [Accessed 5 Nov. 2019].