Measuring Software Engineering

Xiaolei Zhang

December 4, 2021

Introduction

It was not until the 1960s that software engineering emerged and developed as an individual sector of broader engineering. In order to study and discuss how software engineering is measured and assessed, let us first find a universally accepted definition for this term. According to Wikipedia, software engineering is the systematic application of engineering approaches to the development of software, which involves architectural designing, development/implementation, testing, launching and evaluation, and maintenance of computer software in a systematic and strategic way according to its principles [1]. This definition sheds some light on what metrics we should look at to measure engineer's performance and how we can leverage the characteristics and the software engineering daily work and ecosystem to efficiently gather and process these metrics. At the end of this paper, I will discuss the legal and moral issues of collecting, processing, and utilizing this data.

Metrics

Source Lines of codes (SLOC)

As a widely used metric, SLOC measures the size of a software program by counting the total number of lines in the program's source codebase. SLOC can be further categorized into two major types – physical SLOC and logical SLOC.

Physical SLOC is defined to simply be the total number of text lines in the source code files, excluding documentation lines. Logical SLOC in general only counts the executable statements. While logical SLOC seems to be more precise as it is less sensitive to formatting, it is more complicated to measure, as you may have imagined.

To demonstrate the difference between the physical SLOC and logical SLOC, let's take a look at the following code snippets.

Code Snippet #1

```
// 1.0.0
for (int i = 0; i < 5; i++)
{
    System.out.println(i);
}
```

Code Snippet #2

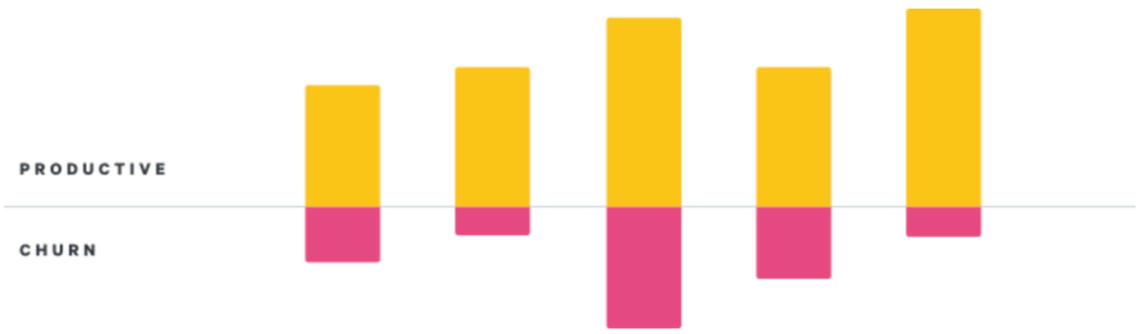
```
// 1.0.1
for (int i = 0; i < 5; i++) { System.out.println(i);}
```

In snippet #1, the physical SLOC is 4 while the logical SLOC is 2. In snippet #2, the physical SLOC is 1 while the logical SLOC remains as 2. Both physical SLOC and logical SLOC ignore the comment line, which is excluded from the counting.

SLOC, especially physical SLOC, provides a straightforward way to estimate the complexity of the codebase, as a metrics of engineering performance. However, as SLOC only counts the lines of codes, it might discourage code reuse and lead to redundant codes and inefficient function calling stack.

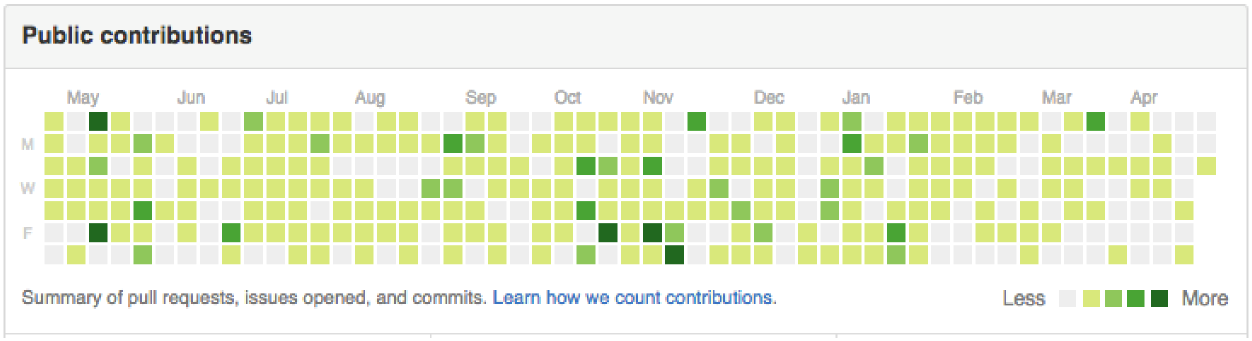
Code Churn

The code churn is defined as the percentage of a programmer's code that is a modification to a previous phase, over a period of time. Concretely, it is calculated by summing up the lines of code that are added, modified, or deleted. Code churning is a normal part of software development, as engineers test out different approaches for problem-solving. It is also dependent on team or project, and the churning would be viewed as normal as long as it aligns with the range by the team or project.



Number of Commits

A commit is an operation which sends the latest changes of the source code to the repository [2]. The total number commits are frequently used to evaluate an engineer's code thruput, reflecting how much changes and codes they have committed. However, as commits have various sizes and imply no change complexity, it might be misleading to be used as a decisive metric.



Summary of pull requests, issues opened, and commits. Learn how we count contributions.

Number of Ticket Close/Bug Fix

A software project is commonly planned with separate steps and is distributed into multiple tasks/tickets which are then assigned to individual engineer(s). These tasks usually contain an estimated finish time or ticket size, dependent and blocking tasks, level of difficulty and etc. Together with these individual tasks, there's also a centralized platform to monitor the completeness and progress of the tasks for individuals. Therefore, the number of closed tickets naturally becomes another metric to be considered while evaluating an engineer's performance. In addition to the simple count of tickets, estimated finish time, or ticket size, level of difficulty can be factored in as well, to better gauge the real efforts of the tasks.

Similar to tickets that are often planned work scope for a project, bug claims are filed during product testing or dogfooding, and are often labeled with estimated time/size and level of urgency. Especially, the level of urgency metric is closely linked with how much the product is negatively impacted in aspects like security, integrity, financial, and latency. Closing of these bugs also showcases the performance and capability of individual contributors, so either a simple count or a more complex summation considering the urgency and sizing may become another matric while conducting the performance measurement.

Number of Change Reviews

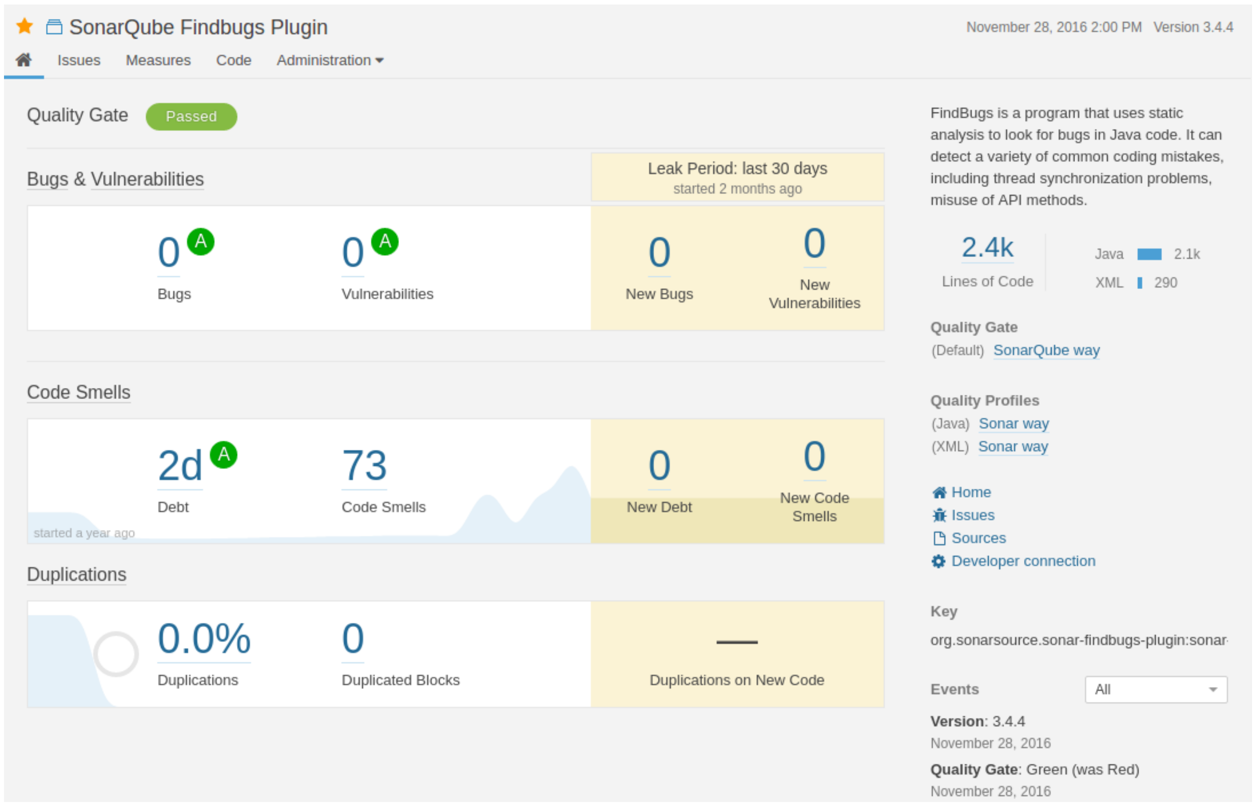
During project implementation, writing clean code with sufficient testing and documentation is certainly crucial. However, what is equally important is the ability to review peers' codes, spot any flaws in advance, improve the call stack, increase the reuse of functions, and in general maintain the codebase to a high standard. These efforts happen during code change reviewing when an engineer leaves comments on another engineer's code change.

Reviewing code changes can take a longer time than could be expected. The reviewer has to take a large amount of time to fill in the context, understand the logic of the approach, evaluate the side effects of the changes, and give advice. Senior engineers often spend more time reviewing than in writing code changes, as they are more familiar with the codebase and have a clearer picture of the overall product. Therefore, it makes total sense to consider the number of change reviews as a significant metric while evaluating engineering performance.

Platforms for Data Gathering and Calculations

SonarQube

SonarQube is an open-source platform for continuous inspection of code quality to perform automatic reviews with static analysis of code for bug detection and security protection[3]. It is currently support in more than 20 programming languages.


SonarQube Findbugs Plugin

November 28, 2016 2:00 PM Version 3.4.4

Quality Gate: Passed

Bugs & Vulnerabilities

Leak Period: last 30 days
started 2 months ago

0 A	0 A	0	0
Bugs	Vulnerabilities	New Bugs	New Vulnerabilities

Code Smells

2d A	73	0	0
Debt	Code Smells	New Debt	New Code Smells

started a year ago

Duplications

0.0%	0	—
Duplications	Duplicated Blocks	Duplications on New Code

FindBugs is a program that uses static analysis to look for bugs in Java code. It can detect a variety of common coding mistakes, including thread synchronization problems, misuse of API methods.

2.4k Lines of Code

Java	2.1k
XML	290

Quality Gate
(Default) SonarQube way

Quality Profiles
(Java) Sonar way
(XML) Sonar way

Key
org.sonarsource.sonar-findbugs-plugin:sonar

Events: All

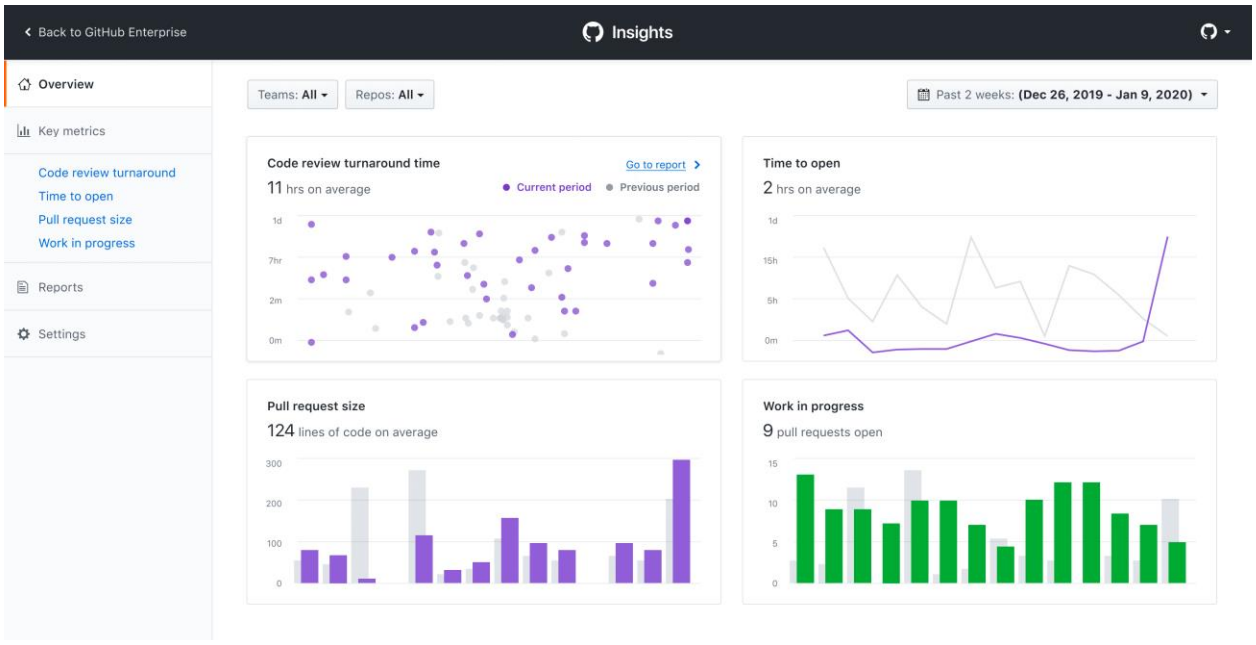
Version: 3.4.4
November 28, 2016

Quality Gate: Green (was Red)
November 28, 2016

Github Enterprise Insights

GitHub, as a widely used tool of internet hosting for software development and version control via Git [4], enjoys its natural advantage to collect engineering development data and offer insights.

Currently, as shown its official promotion page, Github Enterprise provides an analytic dashboard to demonstrate the key metrics of the engineering team, including but not limited to the code review turnaround time, time to open, pull request size and the number of work-in-progress pull requests. It can also involve the basic metrics we have discussed beforehand, like the number of commits and the number of reviews. These metrics could be aggregated by any set of employees over various time periods.

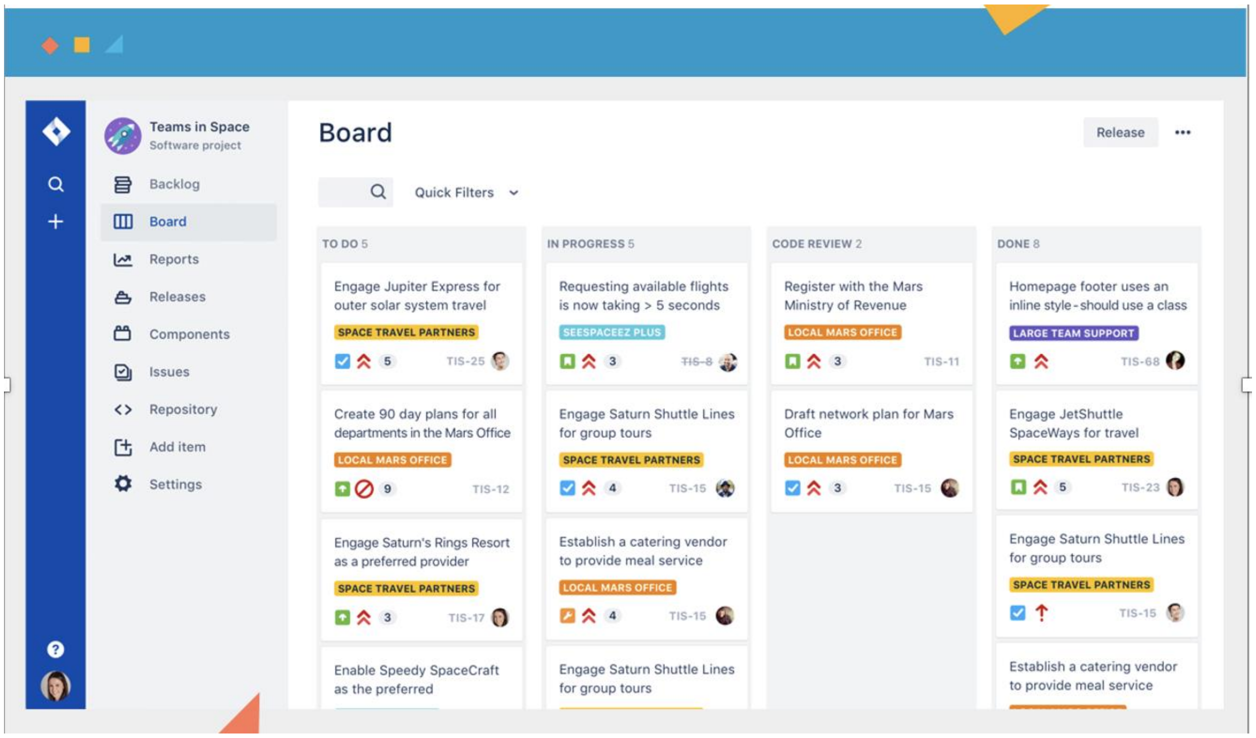


One of the obvious benefits of using Insights is it avoids sending a team's development data to another third-party application if the team has already adopted GitHub as its host of codebase and tool for version control.

Atlassian Jira

Jira, developed by Atlassian, is a proprietary issue tracking product that allows bug tracking and agile project management [5]. It has numerous features for project management, including the road map, tasks tracking inboard or table view, backlog management. These features correspond to some of the metrics we mentioned above, such as the number of tickets closed including their estimated time, level of difficulties, and urgency.

In addition, with Jira, all of these tasks could be easily grouped into a sprint or a scrum, and further split by to-do, in progress, in review, and done. The integration with Jira facilitates the team's project management.



TO DO 5

Engage Jupiter Express for outer solar system travel
SPACE TRAVEL PARTNERS
5 TIS-25

Create 90 day plans for all departments in the Mars Office
LOCAL MARS OFFICE
9 TIS-12

Engage Saturn's Rings Resort as a preferred provider
SPACE TRAVEL PARTNERS
3 TIS-17

Enable Speedy SpaceCraft as the preferred

IN PROGRESS 5

Requesting available flights is now taking > 5 seconds
SEESPACEEZ PLUS
3 ~~TIS-8~~

Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS
4 TIS-15

Establish a catering vendor to provide meal service
LOCAL MARS OFFICE
4 TIS-15

Engage Saturn Shuttle Lines for group tours

CODE REVIEW 2

Register with the Mars Ministry of Revenue
LOCAL MARS OFFICE
3 TIS-11

Draft network plan for Mars Office
LOCAL MARS OFFICE
3 TIS-15

DONE 8

Homepage footer uses an inline style - should use a class
LARGE TEAM SUPPORT
TIS-68

Engage JetShuttle SpaceWays for travel
SPACE TRAVEL PARTNERS
5 TIS-23

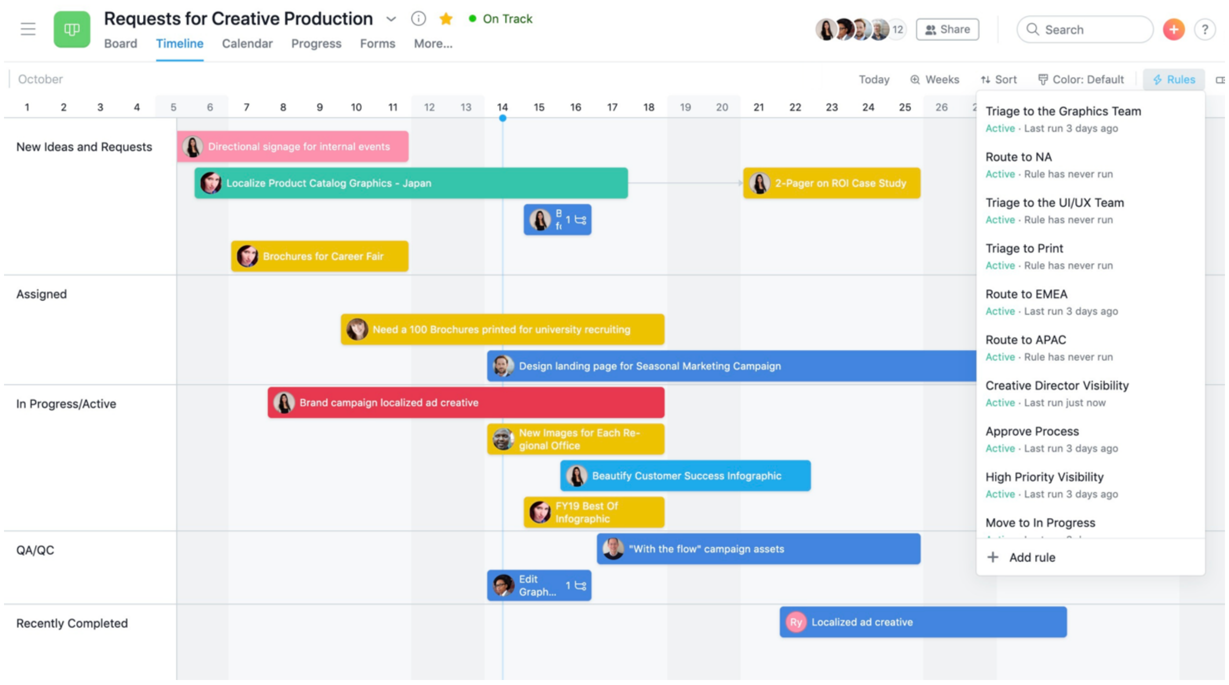
Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS
TIS-15

Establish a catering vendor to provide meal service

Asana

Asana, a similar tool as Jira, is designed to help teams organize, track, and manage their work [6]. Compared with Jira, Asana is not only able to help manage software development, but also supports project management across companies and fosters collaboration. In general, it offers an excellent solution for larger projects and planning, sometimes even at a company level. Because of this feature, Asana is often adopted from top to bottom and gradually becomes a day-to-day tool for smaller engineering teams and projects.

Similar to Jira, Asana incorporates features of task tracking in the list, board, and timeline, and calendar views, supporting user friendly action like dragging, labelling and notification sending.


Requests for Creative Production
On Track
Board
Timeline
Calendar
Progress
Forms
More...
Share
Search
October
Today
Weeks
Sort
Color: Default
Rules
New Ideas and Requests
Directional signage for internal events
Localize Product Catalog Graphics - Japan
2-Pager on ROI Case Study
Brochures for Career Fair
Assigned
Need a 100 Brochures printed for university recruiting
Design landing page for Seasonal Marketing Campaign
In Progress/Active
Brand campaign localized ad creative
New Images for Each Regional Office
Beautify Customer Success Infographic
FY19 Best Of Infographic
QA/QC
"With the flow" campaign assets
Edit Graph...
Recently Completed
Localized ad creative
Triage to the Graphics Team
Active · Last run 3 days ago
Route to NA
Active · Rule has never run
Triage to the UI/UX Team
Active · Rule has never run
Triage to Print
Active · Rule has never run
Route to EMEA
Active · Last run 3 days ago
Route to APAC
Active · Rule has never run
Creative Director Visibility
Active · Last run just now
Approve Process
Active · Last run 3 days ago
High Priority Visibility
Active · Last run 3 days ago
Move to In Progress
Add rule

<u>Other Tools</u>

Measuring engineering performance and facilitating project management and cross-team collaboration is a highly growing market and therefore has many competitive projects, each of which has its pros and cons. Some examples would be Gitlab, Bitbucket, and Monday.com.

Algorithms for Data Computation

<u>Static Code Analysis</u>

Static code analysis automatically detects a potential bug, mis-formatting by scanning through the codebase to ensure basic codebase health. This code analysis is usually programmed on a pre-determined set of rules, and later on, conducted on every single commit.

<u>Machine Learning</u>

Various machine learning algorithms, such as unsupervised clustering and supervised reinforcement learning, could contribute to processing the data and measuring software engineering. As we have demonstrated above, measuring software engineering is complicated with numerous metrics or dimensions, and as well as different considerations. The use of machine learning offers help to aggregate these super multi-dimension data, which can be later turned into a standard algorithm for engineering measurement as a tailored solution for an individual project or company.

Legal and Ethical Considerations

Measuring software engineering is crucial for both individual and team/company evaluation and planning, but an excessive or inappropriate collection of development data borders on the invasion of employee privacy and autonomy.

Most of the metrics are logged with the timestamp, and even with IP address and physical location. Individual contributors can be monitored for their detailed behaviors at any time during working hours. Sometimes, personal activity such as website browsing and private chatting could be logged by the monitor tool as well, which is for sure a potential risk for individual privacy. In addition, misuse of these tracking metrics by managers could result in micro-management and harms engineers' autonomy.

After all, it is important to find a balance between tracking engineering metrics and protecting individual privacy and autonomy, which can be realized by clear guidelines and regulations, as well as transparent communications.

Conclusion

As software engineering grows rapidly nowadays, countless ways and metrics have been formalized to measure software engineering. Additionally, many tools have already been developed to facilitate the development and simultaneously collect the engineering data and provide analytical insights via different aggregation logic and algorithms.

While it is understandable for employers to collect these data for the purpose of appropriate tracking and evaluation, it is important to take data privacy and engineer autonomy into consideration, as well as to communicate transparently and clearly on such data collection to individual contributors.

Bibliography

[1] https://en.wikipedia.org/wiki/Software_engineering

[2] https://en.wikipedia.org/wiki/Commit_(version_control)

[3] https://en.wikipedia.org/wiki/SonarQube

[4] https://en.wikipedia.org/wiki/GitHub

[5] https://en.wikipedia.org/wiki/Jira_(software)

[6] https://en.wikipedia.org/wiki/Asana_(software)