Kai Suzuki
CSU33012
19 November 2020

Measuring Software Engineering

Introduction

Today, there are awesome technologies that change our life more comfortably. Certain of those technologies are based on software engineering and there are a lot of engineers to implement and enhance systems. Through the process the engineers work, we have a question: what is the method to examine the quality of software engineering work? How to assess the work each individual engineer performs? This kind of question is so important these days that this field is relatively new comparing to other fields. Also, the software industry is supposed to grow and the number of people involved in this industry is also considered to increase while the demand for software increases as well. In this essay, we will discuss the measurement of software engineering in terms of the ways of measurement, the platform can be used to gather data, algorithms can be used and ethics of this point.

How Can SE be measured

Regarding measuring software engineering, what kind of aspects should we focus on seeing? There are a lot of components that make software engineering and it is clear that only one of the components does not decide whether an engineer works productively or not. In this chapter, the number of lines of codes, the quickness and peer reviews will be discussed.

The number of codes

The number is apparent for everybody, integer 100 fairly means the same thing for all people. Therefore, some people might insist that the number of lines of codes stands the empathy of an engineer. This view can be advocated by people who are not familiar with programming. However, this method cannot measure the quality of work of the engineer.

The important reason that can be presented to refute the above view is that simplicity of the codes is emphasized in programming. A good example may be found in a case that the 20 lines codes can be replaced by only a one-line code. It takes more time for others to understand the former codes than the latter code. Under this circumstance, it is obvious that the complexity of the codes is not effective and the simple code is easily understood by others and can be easily used again at different times or situations.

It is possible to prove that an engineer has so enthusiastic about a project that he/she manages to implement his/her part with many complex codes. In fact, because software engineering is a team project, the codes are worthless as evidence for concluding that the engineer works hard if nobody can read the code.

Quickness

At any task, the speed of work is one of the most important aspects. In general, it is supposed to be much more pleasant to work faster and this rule can apply to software engineering too. In a project team, if a team member works and implements his/her part quickly, other members usually assess the quickness.

Whereas, as the survey about the measurement of developer performance in Twitter shows (Web1), quickness is not the only factor that determines the quality of a software engineering project. Although the speed of implementations of systems in codes is quick, those codes are not appropriately evaluated if they are complex to understand for others.

Furthermore, the quickness can cause that the teamwork does not go well. To illustrate, a case in point in this respect is that a team project is performed most effectively if the level of each member is relatively the same. Therefore, the fact that the speed of work of a member is quicker than others means the balance of levels in the team is not equal and well. Then it has a possibility to reduce the efficacy of the team project.

Reviews by others

So far, I have discussed how the number of codes and the quickness of implementations have an effect on measuring software engineering. The above two factors detail the measurement from the individual developer's view. However, software engineering is a team project; therefore, reviews by others can be the most important point to measure it.

Reviews by others mean the assessment of the work that a developer in a team implements. If the reviews of others are significantly well, it can mean the project of software engineering produces effectively.

As Javdani et al (ref2) define the agile method that is one of the popular ways in software development as “We are uncovering better ways of developing software by doing it and helping it”, the assessment by others in a team can determine whether the productivity of the team is increased or not.

The platform for gathering data

In the previous chapter, the components that have an effect on measuring software engineering were discussed. Nowadays, a developer of software engineering is one of the most popular jobs and there are a lot of developers in the world. For observing those developers are working well and productively, there are many types of platforms and tools as well. In this section, several tools will be introduced, which include the above points in the previous section.

Git analysis

To perform a project and write codes, Git is a significantly used tool for controlling the version of codes. Developing from this foundation technology for software engineering, there are many types of platforms using Git such as GitHub. These technologies are really useful to measure the quality of developer works.

For instance, Code climate is a Git platform that enables a manager in a team to check the condition of a team project. In those platforms, the work that has been done in a team so far is presented such as the number of commits. In addition, the number of codes that are written newly, that of help for others and that of “churm” which means the codes removed immediately are shown as well. Besides, there is a functionality to check how the codes are simple or complex. With this function, it can be easy to make developers in a team write code as simple as possible.

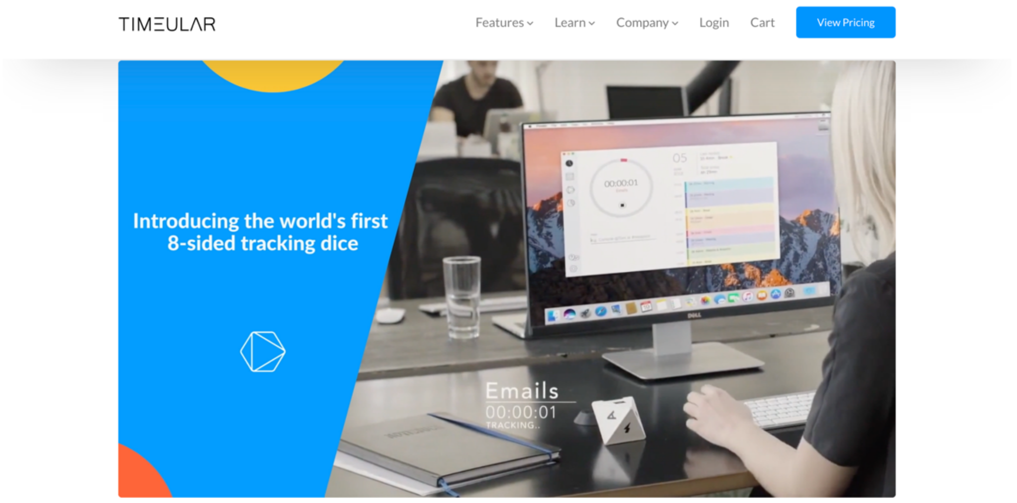
Time analysis

To analyze and improve the productivity of a team in software engineering, it is important for a manager to know what a developer in the team did and is doing. He or she may be lazy in implementing a feature or spend most of the time working on irrelevant points. Otherwise, the developer might not be good at using his or her own time. The job of managers is to control the team to improve the productivity and efficacy in the team. To achieve this job, the manager needs to know the time a developer in the team devotes to an activity.

Timular （Ref3） is one of the tools and technologies that inform the manager of the spent time of a member in a team. For instance, a manager can know how many minutes are spent on email, meeting, talking on a telephone and so on. Therefore, if the manager realizes the time spent of a member is not well-balanced, the manager and the member can consider how to overcome this problem.

From my experience, I check the hours I spent studying for each module and subject every day. By doing this, I can realize whether the study hour is balanced or not and manage myself to enhance the balance of study hours for every subject. This is a personal experience and not about a team project; however, this can be applied to a team project in my opinion.

Admittedly, time is just one factor and it does not determine whether the productivity is increased or not. It is possible that productivity grows in a limited short time and the long hours can cause less productivity. However, time is a concrete factor for everyone; therefore, it can be one measurement of software engineering.



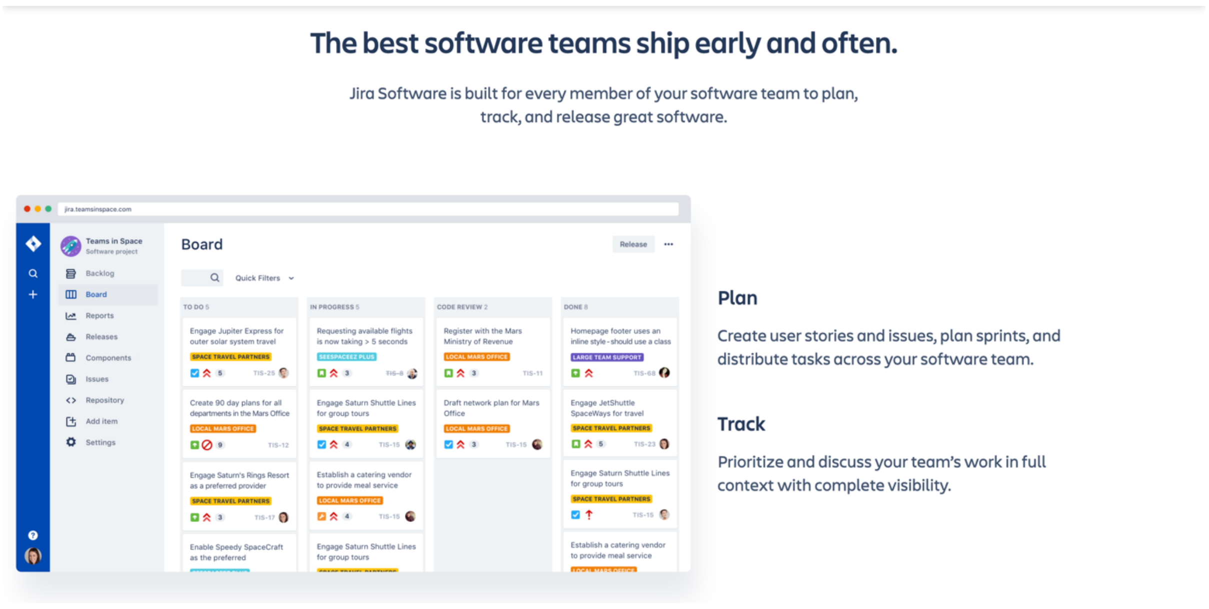
Schedule analysis

For managing a team project, it is necessary to know how the team is operating overall. The tools exist which enable us to see how many tasks are needed to achieve the goal of a project and how many of them have been done so far.

One of the most famous tools is Jira (Ref4) which is easy for anyone to see and understand the management process of a project from the above-mentioned view. In this tool, a task is expressed as a ticket and anyone in a team can add a new ticket.

Also, Jira can show the amount of times that a user in a team logged and used. Therefore, not only developers but also a manager can understand how the task is operating and the difficulty of it.

In addition, Jira is integrated with other technology tools. One of the examples of this is slack. If a member in a team edits and changes the content of a ticket, the notification is sent to other team members through slack. This allows members to communicate with each other and makes the team project operate smoothly.



The best software teams ship early and often.

Jira Software is built for every member of your software team to plan, track, and release great software.

Plan

Create user stories and issues, plan sprints, and distribute tasks across your software team.

Track

Prioritize and discuss your team's work in full context with complete visibility.

Algorithms to be used

In the second chapter, the platforms and tools were introduced for gathering data to manage a team project. The developers in a team are required to write a number of codes in an appropriate and productive time. However, those code themselves are not worthy because, as we discussed in the first chapter, simple and understandable codes should be required to be written. To make this kind of code, an algorithm is necessary to consider.

Halstead Complexity Measures

Halstead Complexity Measures was established as a method to measure software development by Maurice Howard Halstead in 1997. The metrics in the measurement are composed of rules such as

- Mu1 = number of unique operators
- Mu2 = number of unique operators

- N1 = the total occurrences of operators
- N2 = the total occurrences of operands

Also, based on these rules, there are composed rules as well.
- Vocabulary = Mu1 + Mu2
- Length = N1 + N2

and so on.

Function Points

Functions points are viewed as the unit in the same way that hours and minutes are used to measure times. This method is used for estimating and calculating the size and the cost of a software engineering project by setting functional requirements.

This method consists of external inputs, outputs and inquiries and internal-external files. By the use of these components, the assessment of the quality of a software engineering project is calculated. Also, the fact that the data communication is employed and that online data entries or online updates exit, is included by the analysis of the above factors.

Ethics

We have discussed the measurements of software engineering from the views of the methods, platforms for gathering data and algorithms to be used so far. In accordance with the previous topics that were discussed, the performance of a team in software engineering is supposed to be certainly increased. Therefore, in terms of the increase in productivity, these are beneficial. However, at the same time, there can be problems caused by those methods and tools. One of the most problematic matters can be a privacy problem of a developer in my opinion.

The data regarding developers in a team is necessary to manage the operation of the team project. Those data such as times and activities, including mail and meetings, developers are doing are collected and analyzed to enhance productivity. Whereas, collecting these data means observing as many movements of a developer as possible. Therefore, the developer feels being monitored and supervised by a manager.

A case in this point is Timular that is introduced in the preceding chapter as a tool that can record activities a developer performs. As a result, it can monitor the private activities that the developer does not want to pronounce for his/her manager, logging the activities during working time.

In summary, the methods and tools are useful at collecting data of a developer for the increase of productivity; however, they have the possibility of violating the privacy of the developer. Privacy is one of the problems so that other types of problems can exist. Therefore, there are rooms to be improved in measuring software engineering.

Reference

Web1: https://medium.com/@yupyork/the-best-developer-performance-metrics-6295ea8d87c0

Ref2: Taghi Javdani , Hazura Zulzalil, Abd. Azim Abd. Ghani, Abubakar Md. Sultan, On the current measurement practices in agile software development, International Journal of Computer Science Issues, 2012, Vol. 9, Issue 4, No. 3, pp. 127-133.

Ref3: https://timeular.com/

Ref4: https://www.atlassian.com/software/jira