Measuring Software Engineering

Stephen Kinsella

17329337

6th November 2019

Table of Contents

Why Measure Software Engineering:

Before discussing the ways in which the software engineering process can be measured and assessed I want to briefly discuss why it is done. The software engineering process is measured and analysed in order to determine the quality and productivity of the current software development process in order to identify issues and ways in which this process can be improved, it also allows us to predict the quality and progress of future software development projects and it allows for better distribution of workload and priorities between teams and their corresponding members.

It also allows companies to identify the employees within their workforce that can be viewed as a more efficient and better software engineer in order to reward them. Now that I have briefly described the reasoning behind why it is done, furthermore I will discuss the ways in which this process is completed.

Measurable Data:

One of the main ways in which the software engineering process can be measured and assessed in terms of measurable data which is better known as software metrics. A software metric is a measure of software engineering for characteristics of a software engineering project which is measurable and countable. These metrics are useful in numerous use cases such as determining and measuring software performance, measuring the project's productivity, etc.

These metrics can be split into three different types; product, process and project metrics.

Product Metrics:

Product metrics measures the characteristics of the actual software product being developed. The two most important software characteristics that are measured are the size and complexity of the software and the quality and reliability of the software.

Process Metrics:

Process metrics measures various characteristics in relation to the software development process. Characteristics that are measured included are the efficiency of fault detection.

Project Metrics:

Project metrics are used to describe the project characteristics and execution, it measures characteristics such as the number of software engineers working on the software, employee turnover and staffing patterns throughout the software life cycle, the cost of the project, the project's schedule and production.

Internal Metrics

They are numerous types of metrics such as internal metrics that are used for measuring the properties that are viewed to be greater importance to a software developer, one of the most popular methods is lines of code (LOC) measure. Internal metrics have their advantages in the sense that they can be implemented to be completely automated which means it is a cost effective and simple method of measuring software. Internal metrics should be used for large segments of code as it allows a conclusion to be easily drawn on the software. Many measurement tools already exist that allow for automated measurements which do not have to be further inspected, which is a rather efficient method of measuring the software development. However internal metrics are also limited in the sense that their measurements can only be reliable if they are comparing the same language, it would be an inaccurate representation of software development if it was comparing loc of two different programming languages such as C against Python. It is also restricted in the sense that its measurement is dependent on the programming style and the application that is being developed.

External Metrics:

External metrics are a more user-oriented way of measuring the development, it measures software properties such as portability, reliability, functionality and usability.

It measures the software development in terms of performance measurements, the software's quality from the user's perception, response times and completion times.

It is a more subjective form of measuring software development, it also requires both evaluation experts and users to participate at onsite evaluation, this results in it being an expensive method and is not appropriate for cases of distributed users.

Also, unlike internal metric they can't be automated, they results can be automated in terms of analysing the resulting data to form a conclusion. Surveys can be created to collect and analyse the software, but the questions must be handpicked for user-oriented quality characteristics, the weighting and emphasis of some questions are dependent on the author of the survey.

Other Metrics:

Another metric that is used for the evaluation of a software development process is the effort which is expressed as a function of one or more variables such as:

- the size of the program
- the capability of the developers
- the level of reuse

Different models that have been proposed to predict the project cost during the early life of a software life are Boehm's COCOMO model, Putnam's slim model and Albrecht's function point model.

Agile Metrics:

Sprint burndown

Scrum is a software development strategy that organizes software developers as a team in order to reach a common goal. The way in which they record and measure their development is through sprint burndown. Scrum teams organised their development process into time-based sprints. They way in which they are organised is that the team forecasts how much work they reckon can be completed within a sprint. A sprint burndown report then tracks the completion of goals throughout the sprint. The x-axis represents the time while the y-axis represents the work yet to be completed. The goal is to have all anticipated work completed by the end of the sprint. The idea is that a software development team consistently meets its forecast workload by the end of the sprint, however it can tempt software engineers to commit work that is not entirely complete.

Epic and release burndown:

Epic and release burndown charts track the progress of development over a larger body of work in comparison to a sprint burndown, and they can guide development for both scrum and kanban teams. Because a sprint may contain work from several epics and version, it is important to track both the progress of individual sprints as well as epics and versions. "Scope creep" is the injection of more requirements into a previously defined project. Scope creep would ask for new features following the initial requirements have been sketched out for a project. As the team moves through the project, the product owner may decide to take on or remove work based on what they're learning. An epic and release burndown chart keeps everyone aware of the ebb and flow of work inside the epic and version.

Velocity

Velocity is the average amount of work a scrum team completes during a sprint, it is measured in either story points or hours. This is a great way of gauging how long the team will be working on a project. It is also important to use the measurement of velocity against time to see if the velocity of a team grows as relationships and work process begins to be optimized. It can also be used to measure the consistency of a software development team i.e. to measure if their velocity remains the same over time. It also brings development inefficiency to attention if there is a decrease in average velocity.

Time based Metrics:

Leadtime:

Leadtime is how long it takes a software project to go from an idea to delivered software. Teams that take part in the kanban approach favor this measure over velocity as instead of aiming to increase velocity they investigate increasing initiatives which intends to reduce lead time.

Cycle time:

Cycle time is how long it takes for a change to be made to a software system and deliver that change into production. Some teams that incorporate continuous delivery can have cycle times measured as short as in minutes or even seconds in comparison to some teams in which it could take months. This is a good measurement of how efficient a team is at upgrading their existing software.

Open/Close rates:

This software metric measures how many production issues are reported and closed within a specific time period. The general trend of how long it takes for an issue to sorted once it has been reported.

Computational Platforms available to perform this work:

Github:

Github is a graphical web-based interface for a git repository hosting service and is used by millions of software engineers across the world. It contains many features such as version control, access control and many collaboration features. It brings many useful features to the software development process alongside being used for measuring and analysing the software development process.

The number of commits each software engineer can be recorded and analyzed alongside the number of lines of code each contributor adds to the project. It can also be used to record the trends and times in which the project was modified. This can be used to analyze if the project reached its targets in retrospect to time.

However, there are many flaws to recording the productivity of a software engineer through his/her contributions on Github. Even though a developer may have more contributions to a project than the rest of the team does not mean he/she is the most efficient software developer. Large code contribution also corresponds to more bugs as the bigger the project the higher probability of the inclusion of bugs, automated tests in a project does not increase code quality, commit frequency and line output is often higher among junior developers in comparison to senior developers but does not mean they are more productive.

Gitprime:

Gitprime takes advantage of data mining in Git to better visualise team contribution and the productivity of a software project and team. It presents feedback and an insight into many factors of the software development process such as:

- Monitors daily progress
- Tracking of all repositories in a unified view
- Highlights the work pattern of the development team

Gitprime is a beneficial tool to the software development process as it allows managers to overview the entire software process and make what they feel are appropriate changes to increase productivity.

Trello:

Trello is a platform for organising and splitting workload of a team project across it's team members in an easy to use and visual web application. It allows the software management team to allocate work among the team members into many subproblems which can then be marked as completed as the project progresses.
This is a simple and easy way of seeing if project goals are being completed by the software engineers on time. It would also bring to light if the projects workload is not evenly split and if some developers are not reaching their goals as regular as other team members.

Slack:

Slack is a messaging platform primarily used by software development teams as it can be split into channels (for example a software project could be split into front end and back end channels) allowing for less irrelevant messages arising for all team members. Also, private messages can be sent among people in order to share sensitive information and share files in a convenient all in one platform.

Algorithmic Approaches:

Throughout the years many methods have arisen to measure software metrics, but these measurements are useless if they have nothing to be compared with in order to reach an objective conclusion. As software metrics are now an incredibly important part of the software development process many equations and thresholds have been made to check the former analyzed code against the defined thresholds
Here are some of the most popular defined thresholds and how they should be compared with the analyzed code.

Cyclomatic Complexity (CC):

This software metric is used to calculate and gauge how complex a program is, this metric approximately measures the number of connected regions in a dataflow for a method, with a unique point of entry and exit as the complexity of the method. If the method being tested is too complex, the code becomes incredibly hard to understand. It can be used to measure the complexity of individual functions, modules, methods or classes within a program. The cyclomatic complexity can be used to restrict how complex a module of a program is, and it is recommended that if a module of code exceeds 10 it should be split into a submodule in order to limit complexity during development.
The calculation for the cyclomatic complexity of a program references the control flow graph of the program and is seen as:

$$M = E - N + 2P$$

Where

E = number of edges of the graph

N = number of nodes of the graph

P = number of connected components

The threshold is usually between a minimum of 1 and a maximum of 10.

Putnam Model:

The Putnam model demonstrates the time and effort that is required from a software development team to finish a project of a particular size. It is based on the Rayleigh distribution which is described below. It shows that the optimal staffing of a project follows the Rayleigh curve, which shows that only a few software developers are required at the beginning of the project and the number should increase as the project progresses and reach a peak number of staff which will then begin to decline following the implementation and testing of the finished project. This model can also be used to determine the increase in production costs in relation to the compression of the scheduled time. It states that the development efforts and costs increase in proportion to the fourth power of how much the anticipated delivery has been decreased by. This can be used to analyse if a team is over or under staffed and the increase in effort that would be necessary to reach a new milestone.

Rayleigh-Norden staffing pattern equation:

Another way to measure and analyse the efficiency and success of a software project is by measuring the time that the project is estimated to take and compare it to the actual time it takes for the team to complete the project. One method of evaluating the staffing and schedule estimate is through the Rayleigh-Norden staffing pattern equation.

Following the failure of many software projects as a result of project managers treating the project as an activity being performed by a single team this equation came into existence. The standard way of estimating the time it would take a project to be completed was to take the total number of months to complete a project and divide it by the amount of people working on the project. However, a project is performed by multiple teams with different sizes and working at different stages throughout the project and so the estimate was often incorrect. The Rayleigh-Norden staffing pattern equation is seen as a solution to this problem.

The Rayleigh-Norden staffing pattern equation is shown below:

$p(t) = M.t.exp(-t\ 2\ /2td\ 2\ )$

Where,

$p(t)$ = staffing rate at time t

T = time measured in years from the beginning of the project,

M = Maximum staffing rate

td = Full scale development time

M can be derived insuccessive steps as

Ed = Full-scale development effort in man-years

$K = Ed\ /0.39345$

$M = K/td$

Following the estimation of the time taken to complete the project it is then later compared to the actual time taken to determine if the team were more or less efficient than prior estimated.

Ethical Concerns

Even though the measuring and analyzing of the software engineering can bring many benefits such as the recognition of particularly good software engineers, estimation time of delivery, staffing required and the complexity of software it can also arise a multitude of ethical concerns.

Invasion of Privacy:

One of the major issues that has arisen over the last few years is the invasion of people's privacy regarding technology and the way in which their data is stored and analyzed. Is it morally correct to measure and record the actions of a software engineer throughout his working time? It is incredibly invasive to know what an employee is doing for every minute of every working day. It other to analyse this sea of information and data it must be compared to other employees in the workforce or even in the world.

A software engineer should be entitled to complete his work on time and not be criticised for the times and trends in which it was completed.

Sale of Data:

The sale of data obtained by tech companies has become more of an evident issue in the last few years, even though laws such as GDPR and CCPA have been incorporated to tackle this issue they still not solve the issue of the sale data that has been legitimately obtained from the user. This issue can be seen as one that also arises in the measuring of the software development process as the engineer's data is constantly being collected, analysed and compared with other data to conclude the efficiency and productivity of a particular developer. It brings the question of whether it is okay to sell this data to other companies to broaden their data in order to have a more comprehensive and in-depth comparison. I feel it is not evident issue at the moment but we still must be attentive of how our data is handled once it is collected.

Conclusion:

In conclusion there are many benefits to measuring and analyzing the software development process such as the ability to get an accurate estimate of how long a project will take to complete alongside the ability to make real time management decisions based on information learnt from platforms such as gitprime to try maximise productivity in the project. Also, there are numerous ways to measure and analyze the development process so all projects can be accommodated for. I feel like a project can be truly enhanced by taking advantage of the benefits of incorporating the uses of measuring metrics throughout the project to see if goals are met. Even though some metrics can be misleading such as code-based metrics such as LOC that does not mean they can't be beneficial as there are numerous different methods that can be used to analyse software. I believe that now there is no real concern in relation to ethical issues for measuring and analyzing the development as long as companies storing the data measured are responsible with the data they collect. I feel that companies can greatly benefit as they can show and prove to their employees that they strive for a certain standard of quality and efficiency which is a great way of encouraging their team members to meet these standards.

It is often argued that software development is more of an art than a science and so is too complex to measure and analyze but using computational platforms and algorithmic approaches I feel it can be accurately measured and can bring great benefits to the field of software engineering.

Bibliography:

Financesonline.com. (2019). GitPrime Reviews: Pricing & Software Features 2019 - Financesonline.com. [online] Available at: https://reviews.financesonline.com/p/gitprime/ [Accessed 6 Nov. 2019].

GitPrime Help Center. (2019). What Is GitPrime Exactly?. [online] Available at: https://help.gitprime.com/general/what-is-gitprime-exactly [Accessed 6 Nov. 2019].

Grinnell, R. (2019). The ethical use of data. [online] CSO Online. Available at: https://www.csoonline.com/article/3387951/the-ethical-use-of-data.html [Accessed 6 Nov. 2019].

Hackernoon.com. (2019). 7 Rules to Track Software Engineering Metrics Correctly. [online] Available at: https://hackernoon.com/how-to-use-and-not-abuse-software-engineering-metrics-3i11530tr [Accessed 6 Nov. 2019].

Ieeexplore.ieee.org. (2019). Algorithm analyzer to check the efficiency of codes - IEEE Conference Publication. [online] Available at: https://ieeexplore.ieee.org/document/6122740 [Accessed 6 Nov. 2019].

Radigan, D. (2019). Five agile metrics you won't hate. [online] ATLASSIAN Agile Coach. Available at: https://www.atlassian.com/agile/project-management/metrics [Accessed 6 Nov. 2019].

Scet.berkeley.edu. (2019). [online] Available at: http://scet.berkeley.edu/wp-content/uploads/Report-Measuring-SW-team-productivity.pdf [Accessed 6 Nov. 2019].

Stackify. (2019). What are Software Metrics? Examples & Best Practices. [online] Available at: https://stackify.com/track-software-metrics/ [Accessed 6 Nov. 2019].

TechBeacon. (2019). 9 metrics that can make a difference to today's software development teams. [online] Available at: https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams [Accessed 6 Nov. 2019].

The Ultimate Visual Studio Tips and Tricks Blog. (2019). Code Metrics – Depth of Inheritance (DIT). [online] Available at: https://blogs.msdn.microsoft.com/zainnab/2011/05/19/code-metrics-depth-of-inheritance-dit/ [Accessed 6 Nov. 2019].

Virvou, M. and Nakamura, T. (2008). Knowledge-based Software Engineering: Proceedings of the Eighth Joint Conference on Knowledge-Based Software Engineering (Frontiers in artificial intelligence and applications ; v. 180). IOS Press.

www.javatpoint.com. (2019). Software Engineering | Software Metrics - javatpoint. [online] Available at: https://www.javatpoint.com/software-engineering-software-metrics [Accessed 6 Nov. 2019].