Aoife Kettle

CS3012

20/11/18

A report on the measurement and assessment of the software engineering process

The software engineering process can be measured and assessed in many ways. This report will discuss software process measurement in terms of measurable data, the computational platforms available for use, along with the algorithmic approaches used. The measurement process will also be discussed in terms of the ethical nature of the analytics and the social impact of the measurement. In the ever-evolving world of data analytics, the question is no longer what can we measure and record, but rather, what should we?

What data adds value to the process and is ethically correct to record?

The software engineering process is defined as, 'the process of dividing software development work into distinct phases to improve design, product management, and project management. It is also known as a software development life cycle. ... Most modern development processes can be vaguely described as agile.' (En.wikipedia.org, 2018)

The software engineering process can be measured from two points of view, as a development cycle measuring the quality of the process. It should also be accounted for that the engineers involved in development are integral to the process. Human capital is a hugely valuable resource in the software engineering process, the ability to measure human effort is vital. These views are closely linked, but must be measured in unique ways.

Measurement is vital, it reflects the agile mentality of the development. A software engineering process focused on continuous improvement of a product, should also make efforts for continuous process and developer improvement.

To measure it we must look at the various methods of measurement, the positive and negative affects they may have on measurement and the ethical, moral implications of measurement. Process measurement is becoming ever more important, the number of software engineers is expected to increase by 17% by 2024 from 2014 (Mazaika, 2018). With a larger software industry and a growing sector for development, the ability of software engineers to measure their process, reliably estimate work and continuously improve is a necessity.

Measurable Data

Appropriate Data

Data on many aspects of the software engineering process can be collected, however it is important that the right data is collected. It should be measurable, appropriate, and have a purpose. Specific data that aids in measuring the process and assessing the outcomes should be chosen. In the current age of technology, it is less a question of what data can we collect, and more of what data is worth collecting. It must be possible to accurately and consistently measure the data that is being collected, either by automating data collection or requiring engineers to record data at company level.

Purposeful Data

All data collected must serve a purpose, whether that be to encourage performance, track group progress, or aid in the planning and management of the process. The purpose of the data should be considered before any measurement tools or frameworks are put in place.

Data that can be measured ranges from, the amount of lines of code written, to the number of commits made, all the way to the amount of times a person moved in the chair as they worked (Marx, 2008). Collecting and measuring data that does not serve a purpose is a waste of resources and may even hinder the process. It is crucial that all data collected is intended to add value to the process.

Implication of data measured

There must also be an appreciation for the effect of the data collection itself on the process. The reaction of those involved in the measurement, is in a large part influenced by their perception of measurement and can have an impact on the relevancy and appropriateness of the data.

Measuring the wrong data can have a negative effect on the quality of software produced. Human behaviour will naturally compel engineers to work to achieve high levels on whatever scale of measurement is used. This may encourage poor coding, or discourage actions that are not recorded, but are important to the process (Marx, 2008).

Methods for measuring data

As above, data measurement can have an unintended impact on the process, therefore all measurement tools should aim to be as unobtrusive, trusted and reliable as possible. The software engineering process relies heavily on human effort, it is the most used resource. Measuring human effort is important, given its importance to the process, the cost of knowledge capital and the fluctuating nature of effort (T. Acuñai, Juristo and M. Moreno, 2006). It is also the most difficult aspect of the process to measure accurately and reliable in a way that does not obstruct the process.

User Judgement

Many of the early tools for measurement of the software engineering process, were based on users inputting data based on their work. Data that is based on user judgement is prone to error. It allows for bias, mistakes and misunderstandings, deliberate or not. This data is not easily measured; however it is one of the few ways to assess engineer's offline contributions to the team effort.

An early effort on process measurement, was the Personal Software Process (PSP). It focused on organisations ability to measure individual developers, and improvements based on analytics (Johnson, 2013). The original book on this process, (Humphrey, W.S., 1995) presented the measurement in a way that made it possible for organisations to integrate it into their processes. This method was highly based on judgement and required large amounts of data to be recorded by engineers. Because of this, the author intended this method as a bootstrapping practise. PSP was to be used to measure certain areas of the process that required attention, and to teach good practise in the software process.

In today's agile methodology, there may be some benefits to adopting this method. Based on the iterative approach to process development, the PSP could be used to measure various aspects of the process during iterations of the software. Allowing engineers to minimise the impact PSP would have on the process if they were to implement the practise across the entire process. It may be useful to measure certain individual aspects of the process at given times. For example in a certain development cycle to implement a software feature, PSP could be used to measure the time spent on earlier code correct. During another iteration for a new feature, PSP could either be used for the same measurement to create a comparison and measure improvement, or to measure an entirely new part of the process. By using single measures of PSP at a time, it can offer a flexible approach. This could lead to small amounts of measurable data and encourage incremental improvements in the process.

Computational Platforms

The most traditional form of process management, creates a measurement of the process by design. The waterfall method that relies heavily on documentation and records produces an accurate picture of the process, the work that was done and by who. However in the evolving world of software development, the waterfall method is slowly losing its appeal replaced by the flexibility of agile methodology.

Repository management

There are many computational platforms available to assess certain areas of the software engineering process. GitHub allows for easy monitoring of commits and coding history of an individual, it can be used to assess and individuals process on their own work (Github, 2018). However, it does not account for many other aspects of software engineering, one engineer's code git history, will not reflect the time and effort that they have spent aiding another engineer. This part of their contribution to the process is ignored in the data that GitHub provides. GitHub is a good platform for managing the software engineering process as is evident in its popularity, and integration into many other platforms. Yet, it falls short of effectiveness in measuring the process.

BitBucket offers a similar use case, good for managing the process, with some features that aid in team development, but does not go beyond GitHub in its measurement of the process. Repository management is a vital tool for the software engineering process. Allowing multiple engineers to work on the same project and facilitates aspect of the process management in an online space using tools like trello (Bitbucket, 2018).

Analytical Platforms

The PSP method of analytics was effective but time consuming and manual. Some efforts have been made to automate part of the process, including the Leap Toolkit. This kit was designed for the engineer, creating a personal repository of process data (Johnson, 2013). However, it was found to be lacking for the agile development approach, being inflexible and requiring significant overhead.

Hackystat

HackyStat is a platform, designed on GitHub that aims to aid in the analysis of the software development process. It takes in raw data and process it, presenting it in a human readable format, for use in analytics and measurement of the software design process. It mainly measures trends over time, analysing the software process as it progresses and allows for comparisons within the process. The Hackystat approach is to take in as much information as possible, to be used in measurements. It is in a way, a brute force approach. Hackystat takes in a vast amount of data, in the hopes of finding trends and patterns to the process (Johnson, 2013).

Hackystat combines the previously discussed styles of measurement. It collects data from both the client and the server, in the way that PSP would collect client data and a repository collects server data. Hackystat also aimed for fine grained, unobtrusive data collection, an effort to improve the PSP's manual effort requirement. Lastly Hackystat made attempts to measure group development alongside the traditional individual development measurement. However the computational based nature of data collection still falls short of fulling measuring the impact that a team dynamic and work style may have on the software engineering process (Johnson, 2013).

Ownership and control of data

The ethical implications of software engineering process measurement, are ever growing. It has become a question not of what we can measure, but what we should measure. It is not a question of ability, but a moral question. How much information we can gather on the process, without infringing on the privacy of those involved in the process? Who owns the data collected, or has control of it? In the case of the Leap Toolkit, the data was owned by the engineer, is a repository they could move to various projects and organisations (Johnson, 2013). In the case of Hackystat, the development team information collected is less clear cut on ownership. Presuming the control of this data remains with the engineers, an engineer can remove their data from the repository. The remaining data is compromised and less useful. Its high level of privacy offered is appealing in the industry, however this limits its use in team development evaluation.

Limits to Computational Methods

A core failing in the above methods for measurement, is that they only allow for software measurement. Many other factors affect the software engineering process, such as distractions, team dynamic, stress levels, even the weather, that are not accounted for. They are also limited by the willingness of participants, both to take part integrating the measurement methods into their process and to provide correct data where required.

Barriers to integration, such as those found with the Leap Toolkit limit the adoptability of the methods. Where there is a large overhead and required commitment to the measurement it is a disincentive to adopt new methods.

There is of course a limit to how much data can be collected, stored and measured. Hence the importance of only measuring the right data. There is also an ethical concern to the storage of data, and the security of the data. When collecting and storing fine grained details on how people work it must not be overly invasive to their privacy.

Algorithm Approaches

The algorithms used in process measurement are dictated by the purpose of the measurement. They must be effective in measuring both the human and software aspects of process. To do this a combination of algorithms are used to achieve a well-rounded evaluation of the process.

In cases where the purpose is to measure the quality of the process, algorithms used will measure the frequency of actions that are expected high quality processes. To measure the work rate of those involved in the process, algorithms should calculate the frequency, amount and relativity of the work that is being done. Many process measurement styles, will include a combination of these approaches. Given the complex and individualised nature of process measurement, algorithms are often best customized to the purpose.

Accuracy of Data

In the collection of data, both analytical and personable, it is of upmost importance that data is interpreted and manipulated to give an accurate portrayal of the results. The algorithmic approaches must process the data and return it in a usable format that does not represent the results in a misleading result. For example, a graph comparing the time spent by two engineers on a similar piece of code, may show only that one engineer took 4 hours, while the other took 12. This data may be interpreted as one engineer being significantly better and coding in a third of the time, or in one engineer putting 8 extra hours of effort in to produce a better quality piece of code. The algorithms used to process the data must show an accurate picture of what is attempts to portray. When using algorithmic approaches, it is important not to over simplify the data and present an inaccurate picture.

Extensive coverage of performance

It is also important that the algorithm approaches offer an extensive coverage of the performance. For the algorithm to add value to the process, it must cover all aspects of performance that it aims to measure. For example an algorithm measuring the process by assessing trends in commits, needs to be able to assess commit history over a long period of time and provide the information with the necessary context.

Ethical considerations of process measurement.

Process measurement, by nature involves recording, processing and storing large amounts of data on individuals and their work. This poses an ethical challenge to measurement as privacy and ownership must be considerations of any process measurement.

Privacy

In the implementation of Hackystat, there were several barriers to entry noted by the developers. An unwillingness to be measured so completely by the software was vocalised by the engineers involved (Johnson, 2013). Many were reluctant to have their data recorded and processed at such a fine level of detail. Given software engineer's acute awareness and understanding of the dangers associated with data monitoring, it is understandable that they are distrusting of such invasive software.

This can in part be attributed to the world of software engineering that has existed for so long where the engineer was the only person who really understood their own job and it was extremely challenging for management to compare the process of one engineer with the process of another. Engineers may be reluctant to change this opinion that their job is beyond comprehension and measurement by others.

Control of data

The line between the software engineering process, and the software engineer is infinitely fine. Where data collected turns from process measurement to engineer measurement is debatable and almost impossible to define. Here lies the issue with fine grained data collection and the distrust that is seen in industry.

In part it can be handled by the purpose of the measurement, if the intent is to measure the process as a whole, then all data can be anonymously collected, as is the case in the leap tool kit. However for platforms that assess the work of individual engineers control stays with the engineer, for example, Hackystat (Johnson, 2013).

The ethics of measuring the software process are important for several reasons. For the people whose process is being measured it is important that an accurate picture of their work is captured in the measurement process. It is also important that the data collected on them is used and stored in the correct manner. For their privacy and safety this data must only be used to improve the software performance process.

It is debatable as to whether an evaluation of the software process, is a good basis to judge a software engineers work. Given the shortcomings of many of the methods of measurement mentioned above, I do not believe we have the measurement capabilities to capture the entire impact that an individual may have on the process.

Hackystat goes some way to relieving the concerns of software engineers that their data will be misinterpreted, given that it offers trends over time, rather than individual points of data. This means the whole process is measured, and the effectiveness of the team can be measured. By comparing teams in this manner, it is possible to, in some way, account for the impact of individuals whose work was not accurately reflected using other methods of measurement.

Conclusion

Measuring the process of software engineering is a challenging task. It requires a combination of metrics and approaches to achieve a full and accurate picture. There are many varied methods to measuring the software engineering process.

From the observations above, is appears to be wise to measure the process incrementally as the PSP outlines and with a clear goal. Measurement for the sake of measurement is not only a drain on resources, but may unnecessarily infringe on privacy and the process. It is also important to strike a balance between sufficient measurements for improvement that does not hinder the process. The leap toolkit is a helpful tool for developer's personal development and combined with the theories of PSP may be a valuable resource.

As is outlined here (Humphrey, W.S., 1995), a core part of improving the software engineering process, involves improving the software engineers approach, and commitment to the task. Many of the measurements above have difficulty measuring the group process, for practical and ethical reasons. The real value in the measurements lies in the giving engineers the personal ability to measure and improve their personal process.

References

Johnson, P. (2013). Searching under the Streetlight for Useful Software Analytics. IEEE Software, 30(4), pp.57-63.

MArx, D. (2008). Lines of Code and Unintended Consequences. [Blog] DUSTIN'S SOFTWARE DEVELOPMENT COGITATIONS AND SPECULATIONS. Available at: https://www.javaworld.com/article/2072312/lines-of-code-and-unintended-consequences.html [Accessed 08 Nov. 2018].

Mazaika, M. (2018). Will The Demand For Developers Continue To Increase?. [online] Forbes. Available at: https://www.forbes.com/sites/quora/2017/01/20/will-the-demand-for-developers-continue-to-increase/#3a1c256233ee [Accessed 08 Nov. 2018].

Humphrey, W.S., 1995. A discipline for software engineering. Addison-Wesley Longman Publishing Co., Inc.

En.wikipedia.org. (2018). Software development process. [online] Available at: https://en.wikipedia.org/wiki/Software_development_process [Accessed 10 Nov. 2018].

T. Acuñai, S., Juristo, N. and M. Moreno, A. (2006). *Emphasizing Human Capabilities in Software Development*. [ebook] IEEE Computer Society, pp.94-101. Available at: https://s3.amazonaws.com/academia.edu.documents/46862076/ms.2006.4720160628-12305-1brwqqk.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1542740037&Signature=FaGSUL4LIK5L%2BDruTPfFiPWkwGQ%3D&response-content-disposition=inline%3B%20filename%3DEmphasizing_Human_Capabilities_in_Softwa.pdf [Accessed 12 Nov. 2018].

GitHub. (2018). *Build software better, together*. [online] Available at: https://github.com/ [Accessed 10 Nov. 2018].

Bitbucket. (2018). *Bitbucket | The Git solution for professional teams*. [online] Available at: https://bitbucket.org/product [Accessed 10 Nov. 2018].