Measuring the Software Engineering Process

Daniel Di Mascio - 17333014 - CSU33012

Contents

Introduction

"Software engineering is a detailed study of engineering to the design, development and maintenance of software. Software engineering was introduced to address the issues of low-quality software projects. Problems arise when a software generally exceeds timelines, budgets, and reduced levels of quality. It ensures that the application is built consistently, correctly, on time and on budget and within requirements. The demand of software engineering also emerged to cater to the immense rate of change in user requirements and environment on which application is supposed to be working."

- The Economic Times

In this assignment, I was tasked with considering the ways in which the software engineering process can be measured and assessed under four key headings:

1. Measurable data
2. Computational platforms available to measure data
3. Algorithmic approaches available
4. Ethics & concerns surrounding this kind of analysis

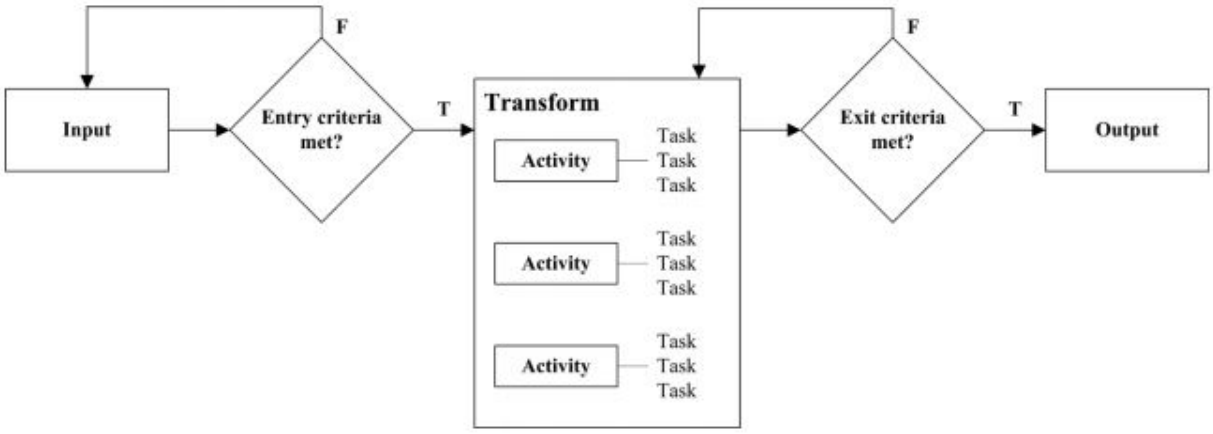
As mentioned above in the quote from The Economic Times, software engineering was first introduced to address a number of issues plaguing the early days of software developers. Developed software could often be of low and unreliable quality, due to factors ranging from budgetary to time frame issues as well as being inconsistently coded.

Software engineering aims to ensure products are developed on time, within budget, suit the end user's needs and coded consistently. The software engineering process is what achieves this goal.

The Software Engineering Process

The software engineering process is a set of guidelines, broken into four key steps, that facilitate the creation and development of software. Regardless of the project being carried out, these four steps should be followed to ensure a successful product:

1. Software specification
2. Software design & implementation
3. Software validation & verification
4. Software maintenance



(SWEBOK V3.0- IEEE, 2014)

Each of these activities contains many sub-activities within them and can be repeated as many times as necessary to achieve the end goal, but all development processes contain each of these activities in some shape or form.

A software product is judged on its user-friendliness as well as how many features it offers the user. The product is graded in these three areas:

Operational - how well a product operates in areas such as budget, usability, dependability, security.

Transitional - this area comes into effect when a product is shifted from one operating platform to another: portability, adaptability, etc.

Maintenance - this area judges how well a product adapts to the changing world: maintainability, scalability, flexibility.

Software engineering often begins with a user request for a specific task or goal to be achieved, after which the software process begins.

Measurable Data

There are many ways to measure the software engineering process (henceforth called the software process for readability), but none of these processes is considered the outright "best" or "worst". This is because it is extremely difficult to establish a single, or even multiple, measures that can accurately assess the overall skill and ability of a software engineer.

Furthermore, it would seem as though many developers do not consider the collection of data important, compared to coding itself (Sillitti A. et al, 2003). Being a very statistically-minded student, I wholeheartedly disagree with this frame of thought. Gathering data on software engineers and their processes can be of great benefit to software development companies and the software process as a whole. Metrics such as the ones explained below can be used to identify, track, prioritise and communicate any issues, which leads to better productivity as a whole. They can also be used by teams to summarise the status of a project and pinpoint errors arising within one.

This sort of software process data collection was brought into the world around the time of the NATO Science Committee's talks on the idea of software engineering in 1968/69 (Software Engineering, 1968), but initial efforts were laced with certain problems.

First of all, it was difficult for companies to know what kinds of data should be collected. Currently, measures in engineering can be split into two kinds: product metrics and process metrics. **Product** metrics focus more on the product's qualities; its dimensions, physical data, etc. They are used for tracking the state of a product in development, for tracing risks and to uncover potential problems. **Process** metrics measure a product's qualities, such as the effort required or production time. This type of metric attempts primarily to enhance the long term process of the engineering team or organisation.

Lines of Code

An early method of data gathering developed was to measure the lines of code typed in a project. This measure, abbreviated to LOC, is laced with problems. There is much dispute around to what constitutes 'one line' of code, as if every line in a piece is counted, this could lead to the inclusion of things such as dead code, comments and spacing lines.

When technology expanded past the days of punched cards and assembly language, lines were no longer limited to 80 or 96 characters and even more problems with this system were introduced as one line of text no longer necessarily meant just one line of code. Unfortunately, this method is still, at times, used today (Stackify, 2017). With no clear

definition of LOC, it is difficult to swiftly compare one project to another. Lacking a standard unit of measurement is a crucial downfall to this method.

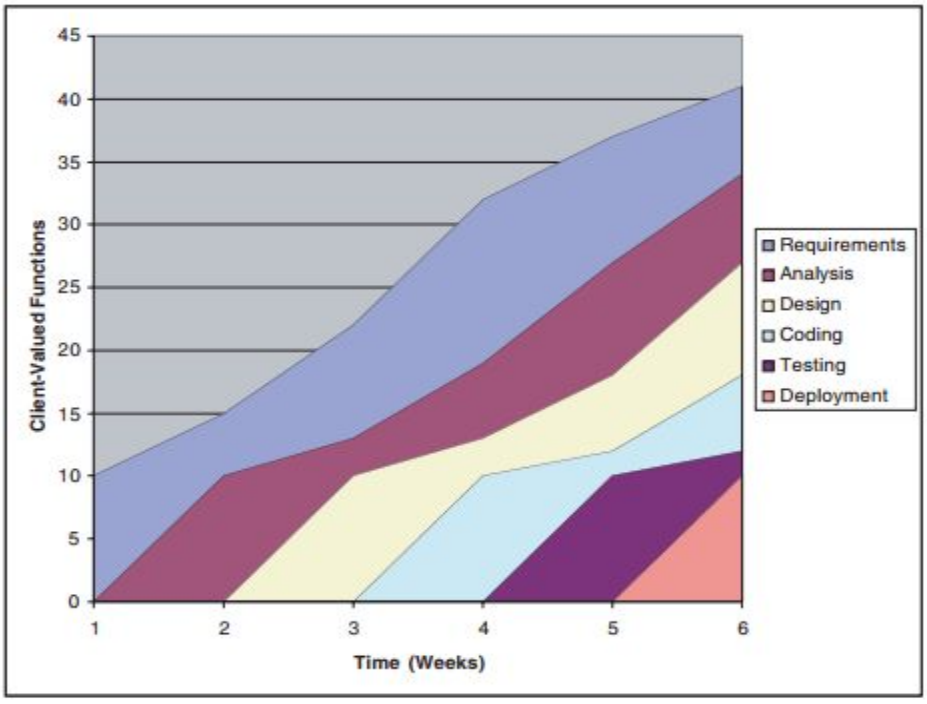
Agile Methods

Agile processes are by far the most commonly used in today's engineering world, with approx. 97% of companies said to use agile methods in their business (State of Agile Report, 2018). This process is named for the way agile teams plan and make decisions. While not directly describing the software, agile methods are invaluable to the development teams of today. One of the main metrics used here is delivered business value (Will Hayes, 2014).

A popular example of an agile method is the Scrum method, which focuses on carrying out a set of prioritised tasks over a period of time called a "sprint", with regular team meetings throughout (The Scrum Guide, 2016).

Lead Time

Lead time is an excellent way of quantifying how long it takes an idea to be developed and presented as software. More formally, lead time measures the time that it takes a unit of V to pass through the development system from its initial input to output (Pearson Education). Lowering a company's lead time improves how responsive its software developers are to the demands of its customers.



(Pearson Education)

Figure 5–2
Cumulative flow of system inventory.

Platforms Available for Gathering Data

Above, I have outlined metrics and techniques that can be used to measure the software process, but platforms for collecting and analysing this data vary.

Personal Software Process (PSP)

The Carnegie Mellon University is an American-based, private research university that in the 1990s developed PSP, one of the first true systems attempting to qualify software development metrics. At the helm of this development was Watts Humphrey, who wrote a book outlining his creation. He believed that a structured development process, as well as an ability to track one's progress, were essential for helping developers to streamline their success. The system was developed with the aim of minimising coding errors and improving the overall software planning process.

Unfortunately, the PSP method requires a developer to manually fill out forms and gauge their own progress, which can lead to incorrect conclusions being drawn (Johnson P, 2013).

LEAP Toolkit

In an attempt to improve on the PSP system and reduce the large margin for error that comes hand-in-hand with PSP, the Lightweight, Empirical, Automated and Portable Software Development Improvement (or LEAP), was born. Leap aimed to automate the data analysis in order to reduce human error, however still required a deal of manual input. LEAP provided many features which PSP did not, such as regression analysis.

As time went on and LEAP came to the forefront of software analysis, it became apparent that there would be no way to fully automate the PSP method of analysis and unfortunately for the PSP way of thinking, automation was the future (Johnson P, 2013).

HackyStat

The University of Hawaii-developed HackyStat took a wrecking ball to the methods of LEAP and PSP by taking a wholly different approach to the concept of data collection. An open-source, server-based programme, HackyStat operates in the shadows. It uses an attachment system whereby it attaches unobtrusive 'sensors' to engineers' development tools,

collecting raw data on a user's work and sending it back to the SensorBase for storage and analysis, which can be accessed by the company as well as other web services for querying.

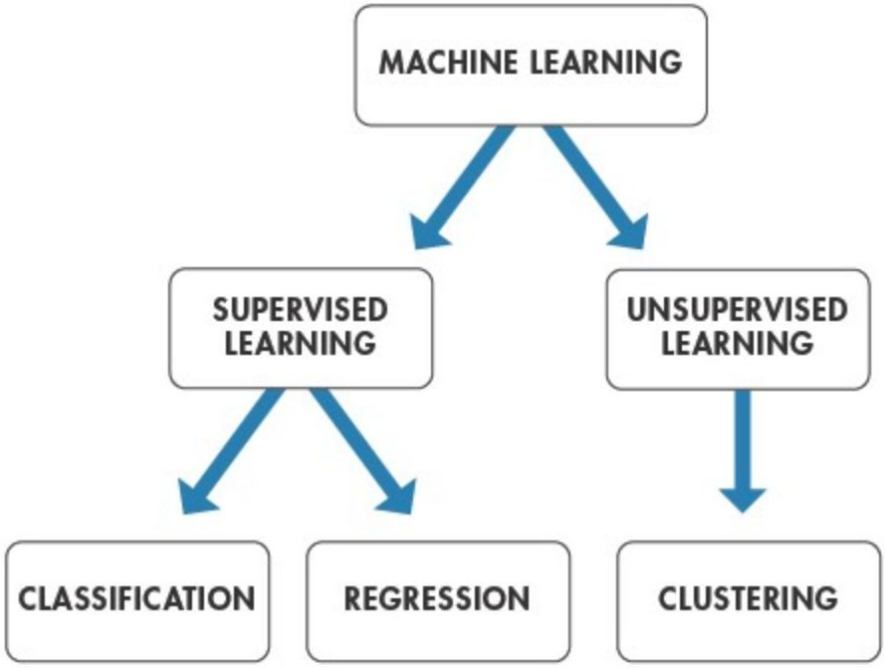
HackyStat takes a big-brother approach, not leaving one stone unturned and focuses on individual data collection by taking measurements of every single aspect of a developer's work. While it has a huge draw to it by the impressive amount of automation it provides, it is often not looked favourably upon due to the sheer wealth of data it collects on its subjects, which can be accessed by one's employer any time they please. (Johnson P, 2013)(Sillitti A. et al, 2003).

This has raised many ethical concerns, which will be addressed in a later segment below.

Algorithmic Approaches Available

The last 10 years have brought about great improvements in computer automation, meaning that the days of manual data gathering for analysis are quickly being left to history. This automation has greatly increased the speed at which the software process can be measured and so in this section, I will look at the machine learning algorithms that have enabled developers to automate this process.

There are two primary types of machine learning: supervised and unsupervised.



Supervised Machine Learning

In supervised machine learning methods, the learning algorithm is given data which is well "labelled" to analyse. Labelled data means that some of the data fed into the machine is already tagged with the correct answer, which the algorithm then uses to predict future inputs. This can be compared to classroom learning, supervised by a teacher who knows the correct answers to the tasks given.

A supervised algorithm learns from training data to help predict outcomes for unforeseen data. Examples

Linear Discriminant Analysis (LDA) - LDA is a classification technique which allows the user to assign multivariate data classification groups in order to reduce its dimensionality, making it easier to process. LDA defines a classification decision bound based on training data and classifies multiple instances of unforeseen data in the set by this bound.

It is a simple, powerful and widely used form of analysis and is widely used in binary classification problems. Unfortunately, LDA is not intended for use in multi-class problems and often becomes unstable if there are few examples or the classes are well-separated (Brownlee J, 2016).

k-Nearest Neighbours (kNN) - k-Nearest Neighbours is a relatively straightforward supervised classification technique which uses labelled data as a basis for classifying unforeseen data by assigning the new data points to whichever labelled group has the k number of nearest labelled data points to it.

The algorithm is highly unbiased in nature and makes no prior assumptions of the data being analysed. The difficulty in this algorithm can lie in choosing the best value for k. Larger values of k tend to reduce the interference of noise found in the data, although at the same time large values can distort the boundaries of the groupings. Generally, a good place to start is by choosing a k which is close to the square root of the total observations. (White A, 2019)

Unsupervised Machine Learning

Unsupervised machine learning methods, as you may expect, do not require prior knowledge of the data, or labels, to perform its analysis. Instead, the algorithm should be left alone to discover information.

An unsupervised algorithm has the advantage of being able to complete more complex processing tasks. However, at times, this method of learning can be more unpredictable than

supervised learning.

k-Means Clustering - Clustering algorithms are a very useful way to visualise data based on their common characteristics. The aim of the k-means clustering algorithm is to divide a dataset into a number of distinct groups, k. The aim for each group is to have similar points inside it whilst observations between itself and other groups are different.

In k-means, a point is chosen and merged with the point closest to it to form a cluster. The next two closest points are then merged together and the process repeats until the desired number of clusters (k) is achieved. As with k-NN, the difficulty may lie in selecting an appropriate k, although there are other statistical methods available to help with this. (White A, 2019)

Machine learning methods such as the ones mentioned above have become much more prevalent over the last number of years and have greatly helped in automating the measurement process. Unfortunately, they do not work extremely well with soft data such as social interaction and thus struggle to dominate the analysis industry as a whole.

Ethics and Concerns

As mentioned above, there are a number of ethics issues surrounding the collection of software development data, as there is with all kinds of personal data. Software such as HackyStat are designed to collect and analyse as much data from a user as possible, however, there are certain kinds of information which many would consider private and the collection of which would be considered intrusive or even immoral. The software itself may not use this data maliciously, but that does not stop the information being misused by either someone with access to said personal information or by someone illegally using this information for personal gain.

A great example of data misuse is that of the high profile case which occurred in 2014 surrounding Uber's "God View". God View is a tool used by the transport company Uber to track not only the locations of its drivers but also the locations of its customers. This tool was reportedly misused by an employee of the company to track the location of a journalist late for an interview with one of Uber's executives. While not available to drivers, this tool was, at the time, supposedly widely available and easy to use at a corporate level. (ObserveIT, 2019)

While this example deals with customers and not the software developers themselves,

it clearly shows how easy it is to misuse someone's personal data (in this case, their live location) without any knowledge or consent from the subject.

Cases such as these are what sparked the release of the General Data Protection Regulation (GDPR) by the EU. These strict laws were brought in in 2018 to replace the existing legislation, the Data Protection Act (DPA) and to better govern the distribution and privacy of personal data. Companies must now ask for consent to hold any person's data and must be able to show a record of this consent. Furthermore, the security measures surrounding the holding of personal data has been greatly increased and the ability to share it greatly restricted (Citizens Information, 2018).

Monitoring employees and storing information on their performance has become commonplace. Unfortunately, it is with great difficulty that one can decide how much data collected is too much. The GDPR regulation has helped greatly in the past year to draw a line in the sand with regards to the ethics of collecting data. I would argue that the concerns of employees regarding the collection and processing of their data are detrimental to a healthy working environment. Hopefully, regulations such as GDPR can help both employees and employers agree on a method of collecting data for the sake of improving company productivity while not collecting too much as to intrude on an employee's privacy in the workplace.

Conclusion

The collection and measurement of data have had a hugely beneficial impact on the software engineering process. Software processes have improved, coding and software development has become more efficient than ever, productivity in the workplace has increased and the number of errors in software products has decreased, all thanks to the measurements put in place by the software process. However, in this world where it has become so easy to automate and forget about these processes, we must be increasingly careful how we implement these technologies which can monitor and measure all aspects of our developers. The ethics and concerns surrounding these processes will require further debate as the speed at which we develop and automate of our systems seems to grow faster by the day.

Bibliography

The Economic Times:
https://economictimes.indiatimes.com/definition/Software-engineering

Silitti A. et al, 2003:
https://www.researchgate.net/publication/4034719_Collecting_integrating_and_analyzing_software_metrics_and_personal_software_process_data

NATO Software Engineering, 1968:
http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

Stackify, 2017:
https://stackify.com/track-software-metrics/

State of Agile Report, 2018:
https://www.stateofagile.com/?_ga=2.104431168.191182351.1573583794-1188359037.1573583794

Scrum Guide, 2016:
https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100

Pearson Education:
http://catalogue.pearsoned.co.uk/samplechapter/0131424602.pdf

Johnson P, 2013:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6509376&tag=1

Brownlee J, 2016:
https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/

White A, 2019:
https://www.scss.tcd.ie/~arwhite/Teaching/STU33011.html

ObserveIT, 2019:
https://www.observeit.com/blog/importance-data-misuse-prevention-and-detection/

Citizens' Information, 2018:
https://www.citizensinformation.ie/en/government_in_ireland/data_protection/overview_of_general_data_protection_regulation.html