Measuring Engineering

CS3012

Software Engineering

Conor Mugan

15319838

Contents

1. Introduction

Software Engineering is a strenuous, ambitious work of art. Every day, masses of software engineers flock to their workplaces and drawing tables in hopes of creating, improving and accomplishing something astounding. A long and arduous process that always has the possibility of coming up painfully short, but can also change the life of any ingenious creator should they endure it. In my opinion, software development projects can never truly be finished, but further improved, revised and completely rebuilt, as the needs of the public grow, along with the aspiration of those developing it. To compete with this continuous upgrading and integration of new ideas, we must also learn how to measure the process of software engineering, through the use of its data and information.

The process of software development and engineering can be easily measure, firstly through the use of data that we can retrieve, analyse and graph. We can use this data to accurately provide a visual representation of productivity of the work force, and also the efficiency and simplicity of our code. This can provide us definitive information, improved through the use of algorithmic measurement approaches and the computational platforms that we use to build it.

2. Measurement Process

Basic forms for measuring the software engineering process would be things such as calculating the amount of lines of code along with the efficiency and functionality of the program and its improvements. Although these may provide a brief, helpful insight into the base level of the project and its development, they do not contribute enough accurate information to us the viewers or those directing the project. This is due to the fact that they can be manipulated to present more impressive results than what is actually present.

An example of this would be spreading simple coding instructions over multiple lines to make the task seem more complex than it actually would be. However, there is more accurate and enticing data that we can collect and scrutinize.

a.) Employee Analysis

Although analysing employee performance and behaviour may seem like an elementary way of measuring the engineering process, it is a vital tool in terms of using its data to further our understanding of the development of the project. Multiple software development companies utilise these techniques, through the evaluation of employee, mid-level and high order data. (Barrett)

Every day, thousands of companies across the world analyse their employees, creating masses of data ranging from the amount of code commits and working hours to more personal issues like their interaction and engagement with fellow members of staff. I feel personal relations within the work place is an important piece of data to examine as a stronger bond between teams of employees can massively improve productivity and overall happiness within the varied levels of the company. (McFarlin)

As McFarlin describes in her article, the main effects of secure bonds are; improved teamwork, employee morale and productivity. One of the main necessities of retaining a healthy mindset and work flow is increasing engagement with others. As the number of friends and associates you communicate with increases, as does your satisfaction in your position and the quality and amount of work you will achieve. This in turn leads to a surge in the retention of the employees and an upturn in the quality and improvement of the software projects that the company develop. We can use this data to incisively measure the process of software engineering from a lower, employee oriented level. (McFarlin)

One class of data that I would like to focus on when considering software development processes is High Order Data. User Entity Behavioural Analysis (UEBA) is the use of metadata and overall monitoring of an employee’s behaviour throughout their working day. (Barrett) As is implied above, work relations affect software projects in all companies, but so does the manner in which employees communicate and carry out their tasks. It is through the investigation of their interactions and computer usage i.e. file accesses, website history etc. that we begin to uncover a deeper level of understanding of the development of a project and the problems that are hampering it. By checking for unusual, destructive behaviour, companies can discover who or what may be impeding the rapid improvement of workload and completion processes in a project. These flaws may then be recognised and dealt with, should need be, leading to a gradual but extremely important recovery and renovation of the work plan.

This High Order Data allows us to visualise the goals and needs of the task at hand, and also the procedures that influence its growth.

b.) Data Metrics

Data metrics are an integral part of software development and best represent measurable data in software engineering as they specify the exact types of data we will analyse and also the quantity of it. A metric is a function applied to a process and thus provides us with multiple results, i.e. data. Although it may seem fairly simplistic, it allows us to obtain consistent, accurate measurements that will further improve our understanding of the processes at hand. Software development companies utilise software metrics in project planning, primarily to improve quality testing, debugging, software optimization and also schedule and budget planning throughout the lifespan of the project. Examples of groups that employ such metrics are the US military and NASA. (Wikipedia, Software Metric)

There are various measurements of software ranging from basic things like lines of code and code coverage to more advanced metrics like connascent software components and Halstead complexity. Code coverage is a modest yet powerful data metric as it provides us with particular data to represent how much of the program's code is actually used and also if it is highly functional. It provides a numeric visualisation of what lines are wasted and useless, and also what is integral to the projects functionality. Despite it being straightforward in its use, code coverage is a clinical component in the foundation of software metrics and data measurement within the software development process.

Invented by Meilir Page-Jones, connascence is the measure of dependencies within a software system. (Wikipedia, Connascence) Connascent software components are exceptionally vital in the development and maintenance processes of software as they can cause massive malfunctions within the system should the threat not be recognised. Two components of the system are connascent if a change in one component requires a similar alteration in the other component. In spite of the fact that connascence may be a predominantly negative effect, it is important for comparing different types of dependencies within software and provides hints at how to improve the overall quality of the software being developed. The measurable data we can collect from this is integral to the development process.

Software and data metrics are identified as spotty in the software engineering world as they are believed to cause more harm than help by certain developers. (Wikipedia, Software Metric) It is perceived that the inaccuracy of certain metrics can cause inaccuracy in the coding and renovation of software as projects are revised to improve the quality of the data measurements that are received. As previously mentioned, certain aspects of the metric measurements can be falsely advertised and changed to match the specified goals of the project, providing an improper claim of accuracy and efficiency. This can lead to an unfulfilled target market and primarily dissatisfactory result. However, it is also argued that these inaccuracies lead to more beneficial improvements as the software is adapted to problems that do not actually exist within the coding. As Tom DeMarco once wrote: "You can't control what you can't measure". (DeMarco) It is those uncontrollable inaccuracies that influence us to strive for improvement and competence.

3. Measurement Tools

Measurable data within the development of software is an integral part of the process, but the platforms and algorithms used to do these computations are even more influential. If it were not for these measurement tools, we could not compute accurate data measurements to improve our software. There also exists ingenious software and gadgets that can measure and compute data about employees and other anatomical related issues in the workplace.

a.) Measuring Employee Data

Humanyze is a company based around helping other businesses get to know their workers and also monitor employee behaviour and collect data. (Barrett) Their Employee Badge Tracking allows them to gather this information and provide the business with in-depth statistics on their engagements with fellow workers and also how each person's body and emotions change based on the situation. As was previously explained, human relations in the work place improves workflow and overall productivity. Regular, healthy interactions with other people is a staple in helping retain an enjoyable and productive working life. The technology that the employee badge uses measures the tone of voice and conversation of every encounter the employee completes. Mapping their daily activities is the basis for addressing issues in the company, whether it be antisocial behaviour or harassment.

It is through the data that the badge collects that businesses can solve their issues and further improve their creativity and efficiency. If an employee is regularly alone then there may be a slight decline in his productivity due to stress, lack of human interaction and further technical problems caused by fear of asking for help. However, if another employee spends an immense amount of time chatting and engaging with others, it can also affect his work flow as he spends more time conversing than actually doing work, possibly leading to a massive drop in group productivity as a whole. This ingenious badge allows those that use it to gage the gap between the two and cut out any dilemmas that the development team may encounter. This invention was introduced due to the fact that executives wanted to "understand what the high-performing branches do differently" in different retail locations. (Heath)

It was discovered that talking and communication was the main solution, proven by the fact that the most successful branches were integrated and team oriented. "The people in the lowest performing branches almost never spoke to each other", the main clue for understanding the source of productivity problems. (Heath)

b.) Measuring Metric Data

There are thousands of different computer platforms that we can use to measure data from metric calculations, ranging from data visualisation workspaces like MATLAB to calculation software for presentations like FlexPro. However, I hove chosen to focus on a rather simple platform as I believe it measures the problems described above in an easy to understand, visually satisfying manner.

Junit Testing is a basic skill for any Java coder to learn whether its for debugging or software enhancements. Personally, I find it extremely easy to use without having to sacrifice functionality or efficiency. Writing test cases to test code coverage is exceptionally easy in this software testing platform as the user can create and modify tests to check that each line of their code runs smoothly and fast. I have used this on countless occasions to further the correctness and power of my software solutions without having to scroll through lines of code and introduce console error messages whenever a bug is encountered.

Code coverage is an important part of data that we should recognise in the software development process as it allows us to visualise how much of our code is wasted, thus decreasing our code value and the profit of our solution. It is also a part of our every day lives as it is a requirement of the automotive safety standard ISO 26262 Road Vehicles guide. (Wikipedia, Code coverage) Another platform that is integral for providing measurable data with respect to code coverage is the Github API, as we can test the efficiency of code along with the lines used using the Maven and Travis builds included in said API.

Junit tests are also critical in calculating, proving and solving code connascence. Connascence is caused by the reliance of one component on another component, meaning if we change one of those units, we will have to solve the issues it causes in the second one. Junit allows us to do this through the use of test cases once again. When we change an integral unit of code whether it be a function or a global variable, it can vastly alter the results that we get and also cause multiple errors and exceptions. Errors can also be introduced due to improper input provided by the user, which can be easily highlighted using tests we have written. The data we gather from this allows us to improve our software to deal with these problems and prevent any further issues from arising.

It is through Junit Testing that we can write special test cases, or improve our code coverage as we introduce new lines to deal with exceptions and reduce the amount of wasted code we thought would be useful before. I quite strongly agree that without Junit testing, it would be more difficult to collect and monitor the process of software development and also the data that it provides.

4. Algorithms

Algorithms are part of everyday work in the software development process. They are used in every computation, every calculation and every chunk of data mining that can be found within the project. We can utilise these results in computing more data, especially to describe the project we are implementing. To do this, we must use further data calculation algorithms.

Data Mining is one of the most integral data algorithms in modern-day computer science. A process that involves uncovering patterns in very large data sets and extracting large data sets using those patterns, to create vivid visualisations of the information to further improve our understanding and knowledge. Data mining can be used on any set of data the user wishes to scrutinize, especially in terms of connasence and lines of code. For example, mining can be used to detect interference between two connascent components in hopes of recognising the dependencies and reducing the interference between them.

The patterns that can occur in software development and the data mining carried out on it is also known as “clustering”. Clustering is the grouping of a set of objects in such a way that the objects in one cluster, are vastly more similar to each other than they are to other objects that are present in a different cluster. (Wikipedia, Cluster analysis) This clustering method can be used in all forms of data mining whether it be the employee data or the metric data.

The data that companies collect about their staff is primarily based around behaviour and relations, but within a software development company, they would also base data around personal code commits and general workflow. Every person would follow a similar pattern in terms of the time periods they commit between which would form a "cluster" that the majority would be part of. However, the development team could measure their inefficiencies and detect weak links in the team that need to strengthened also through the use of clusters. For example, should one team member be slacking and be getting carried by the rest of the team, this can be detected in a cluster to represent the amount of code that has been committed per member. This lazy employee would be represented by an outlying cluster of people who have committed less than the average or required amount of code. An ingenious use of clustering algorithms could be used to prevent such employees lack of dedication and work from occurring, thus maximising the development process.

Machine learning is another massive algorithm based field of computer science that we can use to analyse and modify data during the software development process. Created by Arthur Samuel in 1959, the term 'Machine Learning' explores the use of algorithms that can be constructed to learn from the data we receive and try to predict future data values in the program. It is a very important tool in the software development industry as it can be utilised to devise more accurate time frames for goals to be achieved and to set further deadlines that development teams will have to reach.

This can be done by storing data which is then used to create mathematical models, which can then predict an expected result of the same data provided to the algorithms. I strongly believe that machine learning is an essential part of all software development processes and should be applied to all projects that it can be seen applicable for. (Wikipedia, Machine Learning)

However, there is one slight downfall that exists within the use of machine learning. The scope. The algorithms present in machine learning are coded to work sufficiently for certain sets and types of data, which can impede progress during the development process as the algorithms must be rewritten or modified to deal with different types of data. Extrapolation is whether these algorithms of a certain type may be applied to different data types in different situations. (Wikipedia, Machine Learning) This is the sole significant issue faced when using machine learning in data calculation but can be avoided with the correct coding and perseverance.

5. Ethics

It is easy to say that we can solve all of these issues that exist within the process of software development through the use of employee tracking and implementing machine learning algorithms to improve program efficiency, but is it really that simple? In theory, using a highly technological employee badge to monitor employee behaviour and tracking their day to day habits is a starting point. It allows us to visualise how we may begin to understand the problems that are encountered during the development process, in hopes of reaching an idealistic state where there is strong communication and team work. But we must first look at those effected by this daily monitoring process.

a.) Employee Confidentiality and Trust

Employees in development companies will be the main issue when trying to collect this data, the possibility of them denying the need or want to be tracked continuously throughout their day may seem invasive and unnerving. The employee badge cannot be simply forced around the individuals neck. We must first build a trusting relationship, which may take time and money, possibly more than it would be worth. Generally, people do not like the feeling that their every move is being surveillance, and also may feel it is an invasion of privacy.

One concern I would have regarding the ethics of tracking an employee’s behaviour is their scepticism that a negative personality trait about them will be revealed or personal information. Nobody wishes for their inner emotions and personal space to be disturbed, especially if they are hiding some deeper mental health problems or past issues. I believe that although it could possibly benefit the company and improve the relations between its worker over time, the costs of carrying out such an exercise could hinder their progress. The possible positive outcomes are equal with the negative consequences.

b.) Costs

Finance is another major concern as the time and effort it would take to carry out such research would come with a cost. Including employing people to calculate and manage the data generated by the employee badges, along with the expenses for the purchase and upkeep of the badges themselves, companies may lose more than they will earn. A security manager may also need hiring, to protect the private data collected about each employee to provide anonymity and trust between the employees. A security breach causing the leaking of information could be lethal, ultimately leading to more expenses wasted.

c.) Resource Demands

My final concern would be resources, ranging from the workforce to the data storage systems needed to complete these tasks. Finding the qualified professionals to monitor this would surely cost a lot of money and time to recruit and higher such a high level of labour, without even having to mention the wage bill that would build up. Data storage would most definitely be a major issue, especially with the streams of data that would need to be stored and analysed, along with the processing power to compute such algorithms on massive sets of data. Resources like are already in high demand in the current day, meaning the retrieval of such things to be more of an arduous challenge than would be welcomed.

Overall, I feel that the ethnics of employee tracking and monitoring is a dangerous risk that only the wealthiest of companies should experiment with. It seems like the advantages are overweighed by the heavy, dangerous consequences that could occur should information be accidentally released or accessed. In my opinion, should a software development company take this risk, the rewards could be monumental, an increase in communications and employee relations would surely pay back the faith and expenses that these operations would cost.

Ultimately it is down to the executives to make this decision, the process of software engineering is a hazardous venture full of hardship and stress, that could either be increased or eradicated through the improvement of relations in the workplace. But it would be at a huge cost. However, I strongly believe that the needs and importance of an employee’s mental health should always come first in terms of staff retention and work productivity.

Mental health outweighs physical health to a certain extent and surely the representation that a company values this and wishes to improve it, would most definitely attract aspiring young workers, and possibly even investors, curious about risking their fortunes in hopes of receiving multiple times their investment in return. Furthermore, this would lead to a lot more measurable data we could utilise to monitor the software engineering process and its outcomes.

6. Conclusion

The software development process is a long and ambitious journey, that can be monitored and visualised through the use of the metric and employee data that it generates. The benefits of such a task could be extremely rewarding, but the consequences are equally as catastrophic. Moreover, the time and expenses it costs are a giant risk in terms of the rewards companies will receive which may seem worth it. Every day the sea of data that is calculated deepens, continuously updating and overflowing the statistics, diagrams and graphs that may help us understand the process and the importance of the accolades at the end of it all. There really is no end to a successful development process in my opinion, but the journey is worth the risk. As the founder of ACNielsen once wrote; "The price of light is less than the cost of darkness", and there is no reason to sacrifice experiencing the light, just so we don't have to experience the troublesome, rewarding journey.

References

Barrett, Stephen. "Measurement." 2017. *CS3012.* PDF. 2 December 2017.

DeMarco, Tom. *Controlling Software Projects: Management, Measurement and Estimation*. Yourden Press, 1982. Book.

Heath, Thomas. "This employee ID badge monitors and listens to you at work — except in the bathroom." 7 September 2017. *The Washinton Post.* Article. 5 December 2017.

McFarlin, Kate. *Chron - Importance of Relationships in the Workplace*. n.d. Article. 2 December 2017.

Wikipedia. *Cluster analysis*. 4 December 2017. Document. 5 December 2017.

—. *Code coverage*. 3 December 2017. Document. 5 December 2017.

—. *Connascence*. 3 June 2017. Document. 5 December 2017.

—. *Machine Learning*. 12 October 2017. Document. 5 December 2017.

—. *Software Metric*. 24 October 2017. Document. 11 December 2017.