Measuring Software Engineering

By Luke Feely (17325631)

CS3012

Contents:

Introduction

The objective of this report is to consider the ways in which the software engineering process can be measured and assessed in terms of measurable data, an overview of the computational platforms available to perform this work, the algorithmic approaches available and my opinion on the ethics concerns that are involved with this kind of measurement.

Before we can look at these different ways of measuring and assessing the software engineering process we must first ask, in terms of the engineering process, what is software measurement and analysis? In my opinion it involves the collection of quantitative data for the purpose of aiding the employer/administrator with the current software project and with future decisions in the software engineering process to improve efficiency, quality and understanding. Collecting data on a current software project is necessary for building a working product which meets the functional requirements it was given as well as for tracking its progress. Efficiency can be improved as if the software engineer is being assessed and evaluated, there would be little room for slacking and also allows the employers to asses which employees are most efficient and effective and can help or adjust teams/developers to improve productivity. Measurement allows us to calculative the quality of software projects. Quality will naturally be improved if the software engineer is being assessed. Measurement also enables the employer to figure out who his best employees are and also to let go of ones who are underperforming for the benefit of the project. Measurement and assessment will also help improve our understanding of the software engineering process as it is an incredibly hard process to predict in terms of how successful a project will be and the amount of time projects can take.

As we now understand the purpose of measurement and assessment in software engineering, it is clear that it is crucial in improving the software engineering process. We now look at what is actually measured and the tools available to us to measure and assess this data. However this kind of measurement brings with it ethical issues which we will discuss later in this report.

Measurable Data

Measurement is an ideal mechanism for feedback and evaluation in software engineering. The measurement and information that is fed back to all parties, e.g., developers, managers and customers helps in making intelligent decisions and improving the development process over time. It is therefore crucial that the measurable data we collect is relevant and helpful in improving this process.

There are two main types of measurable data in the software engineering process. Measurements made on the product or the process. Product measurement involves collecting data involving the software itself and how it meets functional requirements and other design criteria, and measurements on the process could be viewed as the collection of 'admin' data involving the development team and progress of the project itself.

Measurements made on the product are more straightforward as the metrics involved are mainly objective rather than surjective. The data returned tends to be absolute values and has little room for interpretations. An example of some of these absolute values would be the total lines of code written, the size of the project, the number of hours of work done and the cost. Cost includes the measure of any resource expenditure used in a project, e.g., staff months, computer time, hardware cost, purchased software calendar time etc.

Other types of data we can measure is to do with the software itself involves testing. Tests are the easiest way to see if the product is meeting its functional requirements and are a crucial part of the software development process. The tests, often written before work is started on the product, test the main functionality of the product to see if it is working as anticipated. The data returned from these tests can be as simple as tests passed and tests failed which identifies to the development team where there are errors.

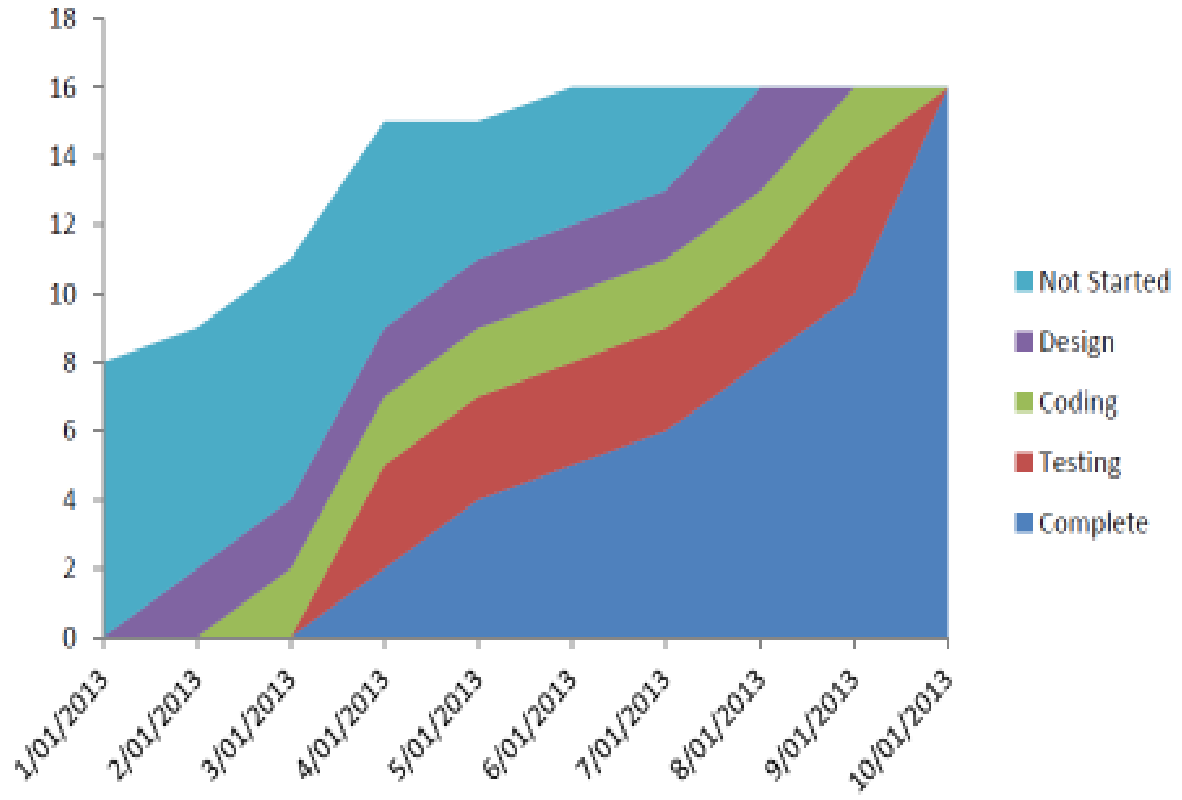
Measurements made on the process and the developer are not always as accurate as they normally don't return absolute values. There are some measurements of course that do return absolute values like hours worked per day of a developer or number of lines of code written but these don't always give a fair reflection of the developers work.

Smoke tests are unlike normal tests as they provide data that establishes the stability of the system. The basic purpose of smoke tests is not to identify faults, but to let the system testing team establish confidence over the stability of the system to start testing the system. This data is very useful for ensuring that the system is functioning and will therefore benefit the software engineering process by providing confidence to the development team.

Productivity is a prime example of 'admin' data collected about the software development process that is very useful to be able to measure. Knowing a software development team's productivity is crucial for predicting future timelines and estimating completion. In agile software development, productivity is known as 'velocity'. Velocity is defined as number of completed user stories (user Requirements) in an iteration. Velocity is only accurate if collected over the long term rather than short time and assumes that the development team stays the same or similar as big changes will affect velocity calculation.

Progress of the development process can be measured using a cumulative flow diagram (CFD). Data collected from a CFD can easily check the status of the project: how much work has been done, what is in progress and how much is still waiting to be done in the backlog. A Cumulative Flow chart helps to gain insight into issues, cycle time and likely completion dates. It is also indispensable for identifying bottlenecks.

(Cumulative Flow Diagram)



The Earned Business Value (EBA) is measurable data that helps track the software development process and gives it a value for a business perspective which is very relevant in software development process. By using EBA, they really want to know how much value the product is currently providing or what percentage of the product is “done.” EBV is common in agile software development. For calculating EBV in a project, a manager “breaks down the project based on the extracted and defined features and user requirements”. Different requirements have different weights. However if there is a high level of uncertainty with the scope of the project, Earned business Value is much less useful.

Computational Platforms

We now have an idea of some of the different types of measurable data we can collect but what we don't know is how this data is collected. In this section, we will talk about some of the different computational platforms that are available to us that help us collect this data.

As there are so much different types of data that can be collected, understandably there are many different platforms available to us with much different functionality. This may also be due to the fact that it is almost impossible to perfectly measure and asses the software engineering process and definitely impossible for there to be a platform that suits all development environments.

When it comes to testing, many IDE's and languages have their own platforms for testing their code. As explained above, unit testing is a crucial part of the development process as it collects valuable data about if the program is running correctly as planned. The platform I know best is Junit. Junit is an open source testing framework for the java programming language. I have used Junit extensively in the last two years for multiple modules and have found it incredibly useful and necessary. Junit makes it easy to write and run tests and it's graphical user interface (GUI) makes interpreting the results even easier. The ability to run multiple tests at once and its simplicity makes it easy to locate and fix bugs which is a crucial and unavoidable part of the software development process

The most common computational platform that collects data on the software development process that nearly every developer uses or has used is Git.

Git is another tool that have become very familiar with over the last year or two and is now a part of my everyday college life. Git is an open source version control system started by Linus Torvald (founder of Linux). It helps developers manage and store revisions of projects. It is most commonly used for code, but it can also be used many other things, like this report! Git is an easy way to track a developer as it tells you how many lines of code they have written or how many commits they have made. A manager of a project can easily look at the repository and see who did what and how much each person did, which may not always be a fair reflection of how hard a software developer has worked. Git's primary function is not to measure and assess the software

development process, but these features are part of it and can be very useful in measuring and assessing the process.

Finally, the last computational platform I want to talk about is cloud analytics.

We know there are many ways to collect data in the software engineering process, but how do we analyse it? Cloud analytics is one way of doing so. “Cloud analytics is primarily a cloud-enabled solution that allows an organization or individual to perform analysis or intelligence procedures” It is also known as “software as a service” or SaaS. It provides this analysis through cloud data warehouses where huge amount of data can be stored. It works like normal data analytics but integrates the service model of cloud computing. It can also be seen as a platform to where software engineers can get the data for their own programs which run their own analytics.

These three different platforms for measuring and analysing data as you can see are very different but all useful in their own way. I choose these three to help emphasise that there is no platform that suits all and does everything. These are just a few examples out of hundreds of platforms available to us, whether they be free or open source, or offer their services as a subscription service.

Algorithms

Collecting data from the software engineering process is the easy part of evaluating a software engineer or product etc. The hard part is actually knowing what to do with the data or understanding what the data tells us. Thankfully machine learning and algorithms are here to help.

Data mining is a process which refers to extracting or “mining” useful information from large amounts of data. Data mining algorithms are used on large amounts of data to discover patters or make important observations that may affect future decisions, or in this case, help assess a software engineering process.

Data mining uses various algorithms to help collect useful information from large amounts of data. Some of these algorithms or techniques are clustering, neural networks and decision trees.

Clustering

Clustering is the process of identification of similar classes of objects. The clustering techniques are used to identify dense and sparse regions in object space and can discover the overall distribution pattern and correlations among data.

Neural Networks

Neural networks have the power to find patterns or trends in complicated and imprecise data that no human or other computer techniques can find. Neural networks are incredibly useful for forecasting needs because of their ability to find trends and patters. For this reason, neural networks are one of the main algorithms involved in computational intelligence.

Decision Trees

A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a decision. It is called a decision tree as it looks like a tree structure which represents

decisions. These decisions generate rules for the classification data. Various tools are used in constructing a decision tree. Decision tree learning is one of the most popular and practical methods for inductive inference.

Ethics

Measurement and assessment is very common in the modern workplace. It is not only in the software engineering process that it exists. It is crucial in improving quality and productivity of workers and projects in any sector of the modern workplace. This growing trend, particularly in the software engineering process, brings with in many ethical issues which need to be addressed. I, someone who only sits in a library with a wall behind him so that no one is looking over his shoulder, believe that measurement and assessment must stick to ethical rules and guidelines that need to be established.

In my opinion the ethics of monitoring in general is a current issue that has received quite a lot of press in recent years. Most notably when Edward Snowden leaked classified information about the NSA and how they have been monitoring EVERYONE'S phones, emails and even webcams without their knowledge or consent. I believe this relates to the most important ethical issue in the analytics of the software engineering process which is that all monitoring and assessment must be transparent. Imagine working somewhere where you are monitored everyday without your knowledge and the amount of ethical issues this raises, including at the forefront an invasion of your privacy. This also brings into question the type of monitoring that an employer should be aloud use. For instance, keystrokes. There is no way that an employer should be aloud have a record of every keystroke and employee makes as this is another huge invasion of privacy. Anything personal an employee does on a work computer becomes public knowledge to their superiors including the more important problem of passwords, pins and other extremely confidential bits of information.

Thankfully the European Commission provides the employee with some protection with monitoring in the workplace. Article 29 of the EU directive establishes a "Working Party on the Protection of Individuals with regard to the processing of Personal Data". This directive ensures that all measurement must be transparent and clear and must respect their privacy by ensuring that all monitoring is fair. I think directives and laws like this are crucial to ensuring the measurement and assessment in the software

engineering process, and others, remain inline ethically and don't cross that ethical line.

Another big problem I see with this measurement and assessment is how this data will be interpreted. The assessment must be fair. A developer who writes :

```
If(true)
{
X++;
}
```

May be considered more productive or harder working than a developer who writes:

```
If(true){ x++; }
```

As the first developer wrote 4 lines of code compared to 1, even though they both do the same thing. We may also see this issue if a company is fond of pair programming where two people share the same workstation and one-person codes and the other supervises and assists. This would lead to the misunderstanding that one employee has done all the work as all the code has been submitted from one user, even though it has been worked on by two. What I am trying to say is that the recorded data may be deceiving and must be properly evaluated. I believe that if it isn't, it would lead to fierce competition and possibly even a drop in the standard of work as employees are trying to maximise how much work it appears they have done.

I am not against measurement and assessment at all as I know it is a part of the software engineering process. I just hope that when I enter the working world and hopefully involved in software engineering process' of my own, that I am fairly measured and assessed and don't have any of the issues or concerns I raised above.

Bibliography

Measurable data:

http://www.cs.umd.edu/~mvz/mswe609/book/chapter3.pdf
https://kanbantool.com/kanban-library/analytics-and-metrics/explaining-cumulative-flow-diagrams

https://arxiv.org/ftp/arxiv/papers/1301/1301.5964.pdf

https://www.qasymphony.com/blog/64-test-metrics/

image:
http://www.clariostechnology.com/images/blog/cumulativeflowdiagram.png

computational:

http://searchsoftwarequality.techtarget.com/definition/JUnit

https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/

https://www.techopedia.com/definition/26516/cloud-analytics

algorithms:

http://iartc.net/index.php/Networks/article/download/56/45

ethics:

https://www.dataprotection.ie/docs/Article-29-Working-Party/u/181.htm