24th November 2017

SOFTWARE ENGINEERING – MEASUREMENT & ASSESMENT

1. ABSTRACT

This report aims to consider the ways in which the software engineering process can be measured and accessed. It aims to do this through addressing the following topics; measurable data, existing platforms available, algorithmic approaches available and finally the ethics of such analysis.

This report will examine existing academic material on the subject of the software engineering process in conjunction with the teachings of Stephen Barrett, a professor of Trinity College Dublin teaching CS3012: Software Engineering.

2. INTRODUCTION

> *"There is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement within a decade in productivity, in reliability, in simplicity." (Brooks F, 1987)*

The first step to understanding software engineering measurement and assessment is to fully define the process of software engineering itself. The notion of 'software engineering' was first proposed in 1968 at a conference held to discuss what was then called the 'software crisis'. (Naur P, 1969). The crisis being that individual approaches to program development did not scale up to large and complex systems.

Software engineering is often described as the engineering discipline that is concerned with all aspects of software production. This is the systematic application of thorough engineering principles to the entire software process including the technical process, project management and tool development.

Software Engineering is a pivotal component in the society at large, it has a huge impact on the way in which we engage with technology on a day to day basis. Being such an important part of society it is important to fundamentally understand how the process of Software Engineering is measured and assessed. The following topic will be discussed in this report under the following headings; measurable data, where to compute such data; algorithms to process the data and finally the ethics involved in such a process. The first step however is to define the landscape of current practice in software engineering process.

3. THE ENGINEERING PROCESS

Agile methods have been used throughout the software engineering discipline since being introduced in the "Manifesto for Agile Software Development", a movement of thought relating to the work process of successful software engineering. At the center of the manifesto is the set of core values that define the efforts, (Cockburn. A et al):

- Individuals and interactions; over processes and tools
- Working Software; over comprehensive documentation
- Customer Collaboration; over contract negotiation
- Responding to Change; over following a plan

This core set of values has brought to light a set of new engineering processes in the field of software development. Among these new set of practices are the Extreme Programming and Scrum methods. These methods can be identified by their iterative approach and adoption of agile methods through the application of the above stated core values.

Extreme Programming revolves around pushing recognized good development practice to extreme levels. As described in the cornerstone book "Extreme Programming Explained", the 4 key values of the process are communication, simplicity, feedback and courage. (Beck, K, 2000). These extreme practices range from pair programming to the story driven architecture of the software, embracing the core agile values of individuals and interactions over processes and tools & customer collaboration over contract negotiation.

The Scrum method again revolves around the iterative agile approach to development. The method is based on the idea of a sprint cycle, a short period of time where work to be done is assessed, priority features are selected for development and the software is implemented and shipped. The product as a whole is broken down into a set of chunks and in the assessment phase of the scrum these chunks are reevaluated and prioritized. This development method can again be seen to incorporate the key agile values of working software over comprehensive documentation and responding to change over following a plan.

It is evidently clear that in the modern age of software development, the agile methodology is commonplace in the industry. In the 11th Annual State of Agile Report, 94% of responding organizations implemented agile methods into their work process. The methods described above are just an example of the agile methods utilized within the industry on a daily basis, they give a brief overview of how the core agile values are integrated into software development.

With the process of engineering defined to be agile, the scope of this report will aim to evaluate the measurement and assessment of software engineering from a framework of agile practice.

4. QUANTITATIVE MEASURABLE DATA

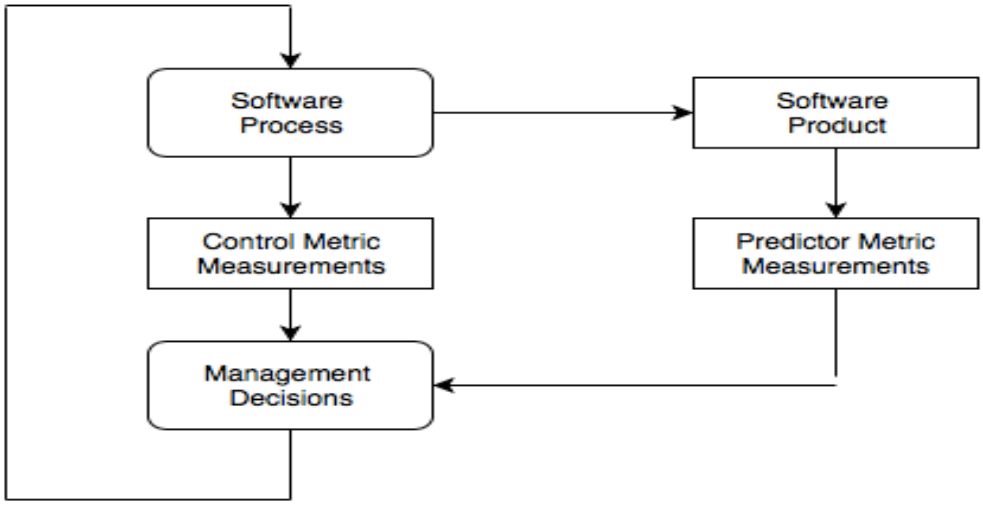
> *"Without adequate quantitative support, we can't readily comprehend, much less master, the complexities of software structure, testability, quality, reliability, or safety. Only with measurement-based frameworks can we approach anything like an engineering ethos in activities vital to effective software production that are, all too often, ill-defined and poorly managed. ", (Offen, R.J, 1997).*

A. Measurement Overview

As described in the book "Software Engineering" by Ian Sommerville, "Software engineering is concerned with driving numeric value or profile for an attribute of a software component, system or product."

In any working field, a project or task will always be measured by time and money. These two basic measurement parameters form the major constraints facing most operating firms today. Resources within a firm are scarce so we must acknowledge and account for the two most fundamental resources when measuring and accessing tasks. It is also worth noting that there is a fundamental link between these two measurement parameters, time is money. With this, we can say that at the bare minimum, measurement of time and money forms a foundational basis for measuring tasks and projects in the field of software engineering.

To add to this foundational basis of measurement in software engineering we must look to specific bases of measurement. As addressed by Pfleeger, S.L et al, in their paper entitled "Status Report on Software Measurement", measurement within software engineering must occur in three key areas, product, process and resources, (Pfleeger, S.L, 1997). I will therefore adhere to these aforementioned bases when defining the measurable data available in software engineering.



The above graphic, (Sommerville, I, 2011) describes the relationship between the measurement bases. Management decisions on resource allocation is dependant on the software product which is in turn dependant on the software process. This intrinsic relationship allows us to define measurement within the framework of these three bases.

When examining these bases in terms of measurement it is important to acknowledge the nature of internal and external attributes of software engineering. External attributes of software engineering are concerned with metrics that interest a large number of entities, these include; maintainability, reliability, reusability and usability. Internal metrics are the inverse; they are metrics pertaining to the state of one entity.

B. Measurement of Product

When we are measuring on a product scale, we can generally split the entities into four defining sections; spec, design, code and test data. These four entities should be present in all good software. Assuming knowledge of these entities, the table below outlines both the internal and external attributes that can be measured.

Entities	Internal Attributes	External Attributes
Specification	Size, reuse, modularity, functionality, redundancy	Comprehensibility, maintainability
Design	Size, reuse, modularity, coupling, cohesiveness, functionality	Quality, complexity, maintainability
Code	Size, reuse, modularity, coupling, functionality, algorithmic complexity, control-flow structure	Reliability, usability, maintainability
Test Data	Size, coverage level	Quality

Product and predictor metrics are generally very hard to define, some key examples of such metrics include; cyclomatic complexity, average length of identifiers and the number of attributes and operations associated with object classes. A basic example of measurement of test data for a product is the number of errors.

C. Measurement of Process

When discussing the process of software engineering we are talking about the steps involved in creating the above stated product. Entities include; constructing spec; detailing the design; implementing code & testing the product. In general terms, internal attributes to measure for work process include time, effort and the quantity of changes and faults to the entities. The table below outlines such entities and their measurable attributes.

Entities	Internal Attributes	External Attributes
Constructing Specification	Time, effort, number of requirements changes	Quality, cost, stability
Detailing Design	Time, effort, number of spec Faults found	Cost, effectiveness
Testing	Time, effort, number of coding faults found	Cost, stability

The number of changes made, time and effort metrics are very easily tracked and defined in software engineering. When discussing the platforms available below this section will be highlighted and expanded upon.

D. Measurement of Resources

Perhaps the easiest of the three bases to measure is that of resource tracking. This basis is not unique to software engineering and as such the metrics defined are commonplace in general management practice. The three key entities for measurement in the software engineering domain are; personnel, tools & environment.

In terms of personnel, key attributes measured include; age, experience and cost. In software engineering the software and hardware tools used to develop software form the entity. Their measurable attributes include; usability, reliability, price & speed. Environment attributes include; size, cost & comfort.

5. PLATFORMS AVAILABLE

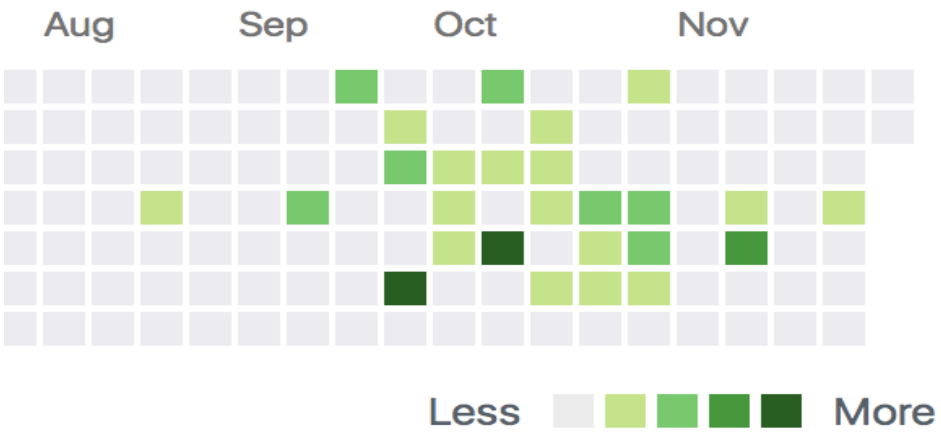
A. Platform Overview

When discussing the platforms available to measure the software engineering process it is difficult to define tools to measure product metrics. In a majority of cases the definition of product metrics is internally defined and differ from product to product. As such there is no general marketplace for platforms that measure predictive, product metrics. Product metrics are usually defined based on the nature of the company and the products that they are creating. The working pipeline used internally will define the basis from which product metrics can be carried out. With this in mind the following section will outline the platforms available to measure process metrics specifically.

B. Platforms for Measuring Process

There is a multitude of platforms available on the market that aim to measure the software engineering process. They provide a product to companies allowing them to efficiently measure the software engineering process at various levels. The key attributes as defined in the above section can be seen to be measured, with the packages aimed and fitted to provide the best measurement service to an agile environment.

Taking GitHub as a primary example of such a platform we can see the power of analysis and measurement available at a free, open source, large-scale level. GitHub is an online environment where software engineers can host repositories containing specifications, design mechanisms, code and everything in-between. It is a web based repository hosting service offering distributed version control and source code management. The platform provides the ability to measure the key process metrics relating to software engineering. Examples of such metrics that are available for collection on GitHub, through the use of the REST API include; number of code commits/churns, number of contributors and frequency of contributor input. These key change parameters are vital metrics relating to the measurement of the software engineering process, the leverage of this platform can be seen by the following image provided on GitHub:



The above heat map details the level of contribution to the GitHub over a period of time, while this visualisation is basic in nature it provides meaningful insight into the level of contribution of certain individuals. For further examples on the leverage of GitHub visualisation available it would be good to reference the assignments submitted as the final piece of work in this module.

Another source of process metrics is the measurement through personal software process structures, PSP. These structures provide engineers with the frameworks to manage their personal workflows and processes on a daily basis. Key parameters involved in PSP include; productivity, reuse percentage, defect information and earned value metrics.

There are a number of platforms available to track such metrics, examples include the SEI platform and the open source project, “The Software Process Dashboard”. PSP places an emphasis on the measurement of defined, accurate and meaningful metrics, (Pomeroy-Huff, 2008). With this information, data analytics can be leveraged to improve the engineering process. Details of such data analysis is defined and further explored in the next section, algorithmic approaches. An example of a dashboard leveraging such analysis is available on the microfocus platform, an application analysing the effectiveness of mobile phone applications, (Microfocus).

6. ALGORITHMIC APPROACHES

A. Process Quality Index

One of the ways in which software engineering is measured is the process quality index as outlined by Pomeroy-Huff et al. The PQI has 5 components outlined in the following table:

Design Quality	Minimum {1.0 , time spent in detailed design divided by the time spent in coding }
Design Review Quality	Minimum {1.0 , 2 times the time spent in detailed design review divided by the time spent in detailed design}
Code Review Quality	Minimum {1.0 , 2 times the time spent in code review divided by the time spent in coding}
Code Quality	Minimum {1.0 , 20/(10+Defects/KLOC in compile) }
Program Quality	Minimum {1.0 , 10/(5+Defects/KLOC in unit testing)}

The PQI is the product of the five elements and will give a value between 0 and 1.0. In general, values below 0.5 will indicate that the product is likely to be of poor quality, with the lower the value the lower the product quality, (Pomeroy-Huff, 2008). This index is a simple algorithmic metric to measure the software engineering process yet it can prove to be very insightful. It leverages data collected from the platforms mentioned above and provides a detailed breakdown of product sections. Through the implementation of this index within product development, managers and team members will be able to access the key areas of success and failure and improve their process going forward.

B. Advanced Data Analytics

When examining any analytical problem in the modern day it is hard to not come across the field of data analytics. Living an an ever increasing technological landscape we can see that data analytics is being leveraged into all aspects of society, from our home to our workplace. Focusing in on the workplace we can see that more frequently data analytics is providing a basis for better quality work output in relation to software engineering. Having discussed the type of data available for collection and the platforms allowing for the data collection we need to fully define and understand the analytical techniques that can be leveraged to gain the most insight out of the data available.

To understand the methods available, we must first define the nature of the data that we have collected. As detailed in the section on measurable data, we defined a number of key attributes that are available to measure. For any single software engineer there are a multitude of variables that are measured using the above stated platforms. As such, when discussing the analytical approach to take when inferring from our data we are discussing multivariate data and techniques as opposed to one dimensional data and techniques.

When dealing with multivariate data there are two key families of techniques that are used; namely clustering and classification. When dealing with these two families of techniques we need to understand the key differences between supervised and unsupervised methods.

According to Brett Houlding, (Houlding B, 2017):

Supervised methods — assume a given structure within the data, e.g, classification, or discriminant analysis.

Unsupervised methods — discover structure from the data alone, e.g., cluster analysis

In general, we can assume that our data will define one of two states, bad engineering practice or good engineering practice. As such, supervised methods that we can use to measure software engineering include; linear discriminant analysis, k-nearest neighbors and logistic regression to name a few. The key algorithmic approach that I think can be leveraged most effectively is discriminant analysis, this application will be explored below.

C. Discriminant Analysis & Classification

Linear discriminant analysis is a classification technique that allows you to leverage your multivariate data with the goal of assigning entities into a classification group. If we are to take for example the process quality index, we can say that the index is accrued from a wide range of variables collected in the software engineering process. At a basic level, we define two groups from our data; bad practice/product & good practice/product.

Linear Discriminant Analysis as such is the process of defining a classification decision bound, say 0.5, and classifying multiple of instances of data by this boundary. This is a fundamental statistical method and is seen to be used in modern practice with machine learning and artificial intelligence.

It is evidently clear that multivariate linear analysis, in particular classification techniques, can be leveraged as algorithmic processes to gain an understanding of the data that we collect in the software engineering process. The advanced analytical methods are used on a wide scale level in process optimization in production and its application can be seen across a multitude of industries from banking to ecology.

7. ETHICAL AND OTHER IMPLICATIONS

Having examined both the data available for collection, the platforms performing collection and the analytical approaches to the data, we must now examine the ethics of such approaches in a working environment. We will approach this topic from a legal point of view and personal opinion on good management practice.

A. The Law

From a legal standpoint, organizations must work to be extremely vigilant when it comes to data protection. There will be the introduction of new EU wide general data protection regulation on 25th May 2018, (dataprotection.ie, 2017). This regulation states that firms in breach of the regulation can be fined the larger of €20 million or 4% of annual turnover. This regulation is causing waves across all industries, companies are moving to combat this extreme risk to the best of their abilities. As such any firm looking to implement measurement over their software engineering process must do so in a prudent and legal manner. Compliance with the regulation is critical.

B. Personal Opinion

Having defined the law in the area of data protection, we know that it is the legal responsibility of the organization to fully protect its internal data. This includes employee specific data, especially relating to performance and input.

From a personal standpoint, I feel that organizations seeking to expand the measurement of their processes need to be extremely tactful with the way in which they collect and leverage data from employees. If general, any appraisal process in a workplace can cause tension between employees and management. When new metrics are being defined and implemented in the workplace it is important to take into consideration the thoughts and opinions of the workforce, failure to do so could result in a bad working environment for all employees.

When people feel that they are being monitored and scrutinized there tends to be a backlash, take for example the recent case of Sgt. Maurice McCabe. He was a garda whistleblower that analyzed and aggregated information relating to bad practices in his workplace. He created a report outlining and highlighting issues he found in the workplace relating specifically to the processing of penalty points. With this report and its presentation to the minister for justice, Maurice McCabe was met with fierce backlash from all levels of the organization, all for the highlighting of key issues in work practice.

To surmise, the implementation of measurement and analytical practices within software engineering must be done in a tactful and sensitive way. Failure to do so could be met with fierce internal opposition

8. CONCLUSION

In brief, this report has outlined the key bases for the measurement of the software engineering process. In terms of agile software development topics of; measurable data, existing platforms, algorithmic approaches and ethical implications have all been addressed. It is clear that the measurement of the process is possible on a number of levels but must be implemented in such a way to avoid the promotion of a bad internal atmosphere.

9. BIBLIOGRAPHY

Beck, K., 2000. Extreme programming explained: embrace change. addison-wesley professional.

Brooks, F. and Kugler, H.J., 1987. No silver bullet (pp. 1069-1076). April.

Cockburn, A et al. Agile Manifesto, http://agilemanifesto.org

Fenton, N, 1999. A Rigorous Framework for Software Measurement, https://www.eecs.qmul.ac.uk/~norman/papers/qa_metrics_article/section_4_framework.html

Houlding, B. 2017. Introduction to Multivariate Linear Analysis. Available at: https://www.scss.tcd.ie/Brett.Houlding/ST3011_files/ST3011slides1.pdf

Microfocus, Mobile App Monitoring. Available at: https://software.microfocus.com/it-it/software/apppulse-mobile

Naur, P. and Randell, B. (1969). Software Engineering: Report on a Conference sponsored by the NATO Science Committee, Garmisch, Germany. 7th to 11th October 1968.

Offen, R.J. and Jeffery, R., 1997. Establishing software measurement programs. *IEEE Software*, *14*(2), pp.45-53.

Pfleeger, S.L., Jeffery, R., Curtis, B. and Kitchenham, B., 1997. Status report on software measurement. *IEEE software*, *14*(2), pp.33-43.

Pomeroy-Huff, M., Mullaney, J., Cannon, R. and Seburn, M., 2008. *The Personal Software Process-SM (PSP-SM) Body of Knowledge, Version 1.0* (No. CMU/SEI-2005-SR-003). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Sommerville, I., 2004. Software Engineering. International computer science series. *ed: Addison Wesley*.

Dataprotection.ie, General Data Protection Regulation. Available At: https://www.dataprotection.ie/docs/GDPR/1623.htm