Measuring Engineering

Liam Greene 15320482

December 6, 2017

Abstract

To deliver a report that considers 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 the ethics concerns surrounding this kind of analytics.

1 Introduction

Software engineering is defined as the systematic application of scientific and technological knowledge, methods, and experience to the design, implement, test, and document software.[1] With the growing prevalence of tech start-ups, software engineering is a growing field. However, with 92% of start-ups failing[2], it is clear that more procedures must be put in place to study and improve this performance. Various measures of data are collected throughout the software development life cycle aiming to continually improve the software process. This data is used in estimation, quality control, productivity assessment, project control, design insight and future decision making throughout a software project. This essay intends to deal with the issue of measuring such an encompassing term to assess the quality of the engineered product or system and contemplate some of the ethical issues that arise from such procedures.

2 Measuring Data

Data measurement is an effective management tool that is composed of five activities: [4]

Formulation:	Performing measurements and developing appropriate metrics for software under consideration.
Collection:	Collecting data to derive the formulated metrics.
Analysis:	Calculating metrics through tools.
Interpretation:	This analyses the metrics to attain insight into the quality of representation.
Feedback:	Communicating recommendations derived from product metrics to the software team.

Software engineering measurements can be subcategorised into direct measures and indirect measures. Direct measures include data that maps information from the empirical world to the formal, relational world[3], such as cost and lines of code produced. Indirect measures include data that we are unable to describe empirically such as quality, reliability, and complexity etc. Indirect measures are often the hardest to successfully gather given their inherent intricacy.

Software engineering measurements can be further divided into software metrics, procurement metrics, production metrics, operations and maintenance metrics, usability metrics and desirable characteristics metrics amongst others. As the below table details, measurement processes are not exclusive to one metric.

Software Engineering Metrics				
Software	Procurement	Production	Operations and Maintenance	Usability
Code Coverage	Return on Investment	First Pass Yield	Downtime	Bugs
Load Time	Cycle Time	Throughput Yield	Spare Parts Needed	Customer Satisfaction
Classes/Interfaces	Quality	Cycle Time	% Spent on Maintenance	ISO
Number of Lines	Cost Savings	On-Time Delivery to Commit	Completion Rate	
Program Size				
Execution Time				

Data collection can be costly, and if the data collection process is too time consuming, it becomes destructive, encouraging the use of metrics that are easy to gather and compute. Therefore, the goal of the data collection should be established with a clear concentration on precision, allowing the information to be focused, direct and accurate, in order to lower the costs of the collection, the storage and the evaluation of the data.

Various forms of data collection have been proposed to fit this information. Dr. Victor Bassili proposed his Goal Question Metric (GQM), a top-down, goal oriented framework for gathering data. The GQM is presented as a seven stop process. [8]

Step 1:	Develop a set of Goals
Step 2:	Develop a set of questions that characterise the goals.
Step 3:	Specify the metrics needed to answer the questions.
Step 4:	Develop mechanisms for data collection and analysis.
Step 5:	Collect validate and analyse the data.
Step 6:	Analyse in a post mortem fashion
Step 7:	Provide feedback to stakeholders

GQM can help in understanding and evaluating an organisation's software engineering practises, while helping to direct the software engineering process due to its emphasis on the establishment of clear goals. However, GQM has been criticised as the selection process of metrics is not clearly defined, and this lack of guidance can lead to bias in metric selection. Technical support for GQM is steadily improving, but not yet perfected.

Orthogonal Defect Classification (ODC) is a process developed by IBM for the collection of data relating to defects in the software engineering process ranging from waterfall to agile software engineering. [16] ODC provides information on the development process and testing process, resulting in an analysis that is more data intensive and computationally based as opposed to human analysis.

Other forms of data measurements exist, and each operation will require a specific method of data collection.

3 Assessing Data

Data analytics is the practice of collecting, analysing, and interpreting data to detect trends and patterns, with the aim of enhancing business productivity and revenue. Once an organisation has decided on their selected data metrics, the method of data measurement and the method of data collection, they must begin to gather and store the data for analysis.

Six Sigma is one of the most popular data-driven approaches for eliminating defects in a process, which has been implemented in software engineering through the use of analysis tools, particularly with the goal of removing software bugs while improving software quality. Six Sigma is comprised of two main submethodologies, DMAIC and DMADV, standing for Define requirements, Measure performance, Analyse relationships, Improve performance, Control performance and Define requirements, Measure performance, Analyse relationships, Design solution, Verify functionality respectfully.

DMAIC begins with boundary analysis and qualitative analysis, in order to collect data and identify the relative problem, then find the cause through metrics and analysis tools. DMAIC is a continuous control activity that should be implemented to improve and optimise data collection, with the purpose to

ensure such problem will not happen in the future.

Six Sigma also has its limitations. It uses statistical analysis tools to find defects in current execution process, but the success of these tools can often be affected by the size the project analysed. The success ratio is higher within small projects when using Six Sigma to manage the project compared to when Six Sigma is used to resolve a problem in a large organization or project.

Data assessment is often associated with the process of gaining an initial understanding of data within the context of the DQAF [data quality assessment framework]. A measurement system analysis (MSA) is a commonly used DQAF. An MSA is an in-depth evaluation of the measurement process that identifies variables in the measurement process through a detailed assessment of the data measurements used. These variables are assessed on five bases: accuracy, linearity, stability, repeatability and reproducibility.

Stability refers to the ability of a project to produce consistent results over time, with a specific emphasis on bugs when applying this method to software engineering. Accuracy compares the true or ideal value to average value produced over multiple tests with the same sample. Linearity evaluates the accuracy over multiples tests using different samples, with the aim of identifying whether the bugs are affected when different variables are changed, or whether the problem remains the same regardless of the variables. Repeatability and Reproducibility survey the precision when using the same appraisers or different appraisers respectfully, as appraisers can test the same program multiple times and receive multiple outcomes.

After examination, the five variables are plugged into a Gauge R&R mathematical calculation or similar, that determines the total variation and specification tolerance and management system error. A general benchmark of 30% is administered, with a measurement system with an error in excess of 30% deemed incapable for usage as the basis of decision making. However, this system is dependent on the organisation's needs. If a company operates with a policy of perfection then an error above 10% could be considered unacceptable, or an organisation that requires a steady product could focus its effort on solely reducing the error in stability at the cost of making further testing more expensive.

By understanding existing measurement systems a team can better understand the data provided by those systems and make better business decisions. [7]

4 Computational Platforms & Algorithmic Approaches to Perform This Work

Computational modules have been developed to improve and streamline the measurement and assessment of data. One such module is the previously mentioned Gauge R&R, which stands for "Gauge Repeatability and Reproducibility", with the purpose of confirming the extent of variation in the data collection process. All forms of data collection will contain some random variance, and Gage R&R studies can prevent organisations making costly measurement errors. There are three main ways to calculate Gauge R&R results, with the most common method identifying the average and range. This method can be simply calculated using spread sheets. The Analysis of Variants (ANOVA) module is the most appropriate when examining the software engineering process, and is calculated using computational software, with the results detailed below:

Variance	Acceptability
Less than 1%:	Acceptable
Between 1% and 9%:	Acceptable depending process cost and application
Greater than 9%:	Unacceptable, process must be improved

Context-aware Software Engineering Environment Event-driven Framework (CoSEEEK) is a framework that provides automated guidance for software developers throughout the software engineering process, combining quality management, process management, context management, and knowledge management while providing support adapted for the needs of the current project situation and CoSEEEK user. The automated data is attuned based on the context-awareness and process-orientation, providing a CoSEEEK user with computational data that relates to their needs. [6]

The Capability Maturity Model Integration (CMMI) is another computational process model that clearly defines what an organization should do to promote actions that improve performance. CMMI was developed at the Software Engineering Institute at Carnegie Mellon University with representation from defence, industry, government, and academia, and is now operated and maintained by the CMMI Institute, an operating unit of CMU. It is the successor of the popular Software CMM. [5]

The CMMI is used to help companies answer the how do we know? questions:

How do we know what we are good at?
How do we know if the process we use is working well?
How do we know if our requirements change process is useful?
How do we know if our products are as good as they can be?
How do we know if our products are as good as they can be?

CMMI is a behavioural model that is comprised of "Process Areas", that are to be adapted to the behaviours of the company, that explain the processes

a company utilises and the behaviours that need to be defined given these processes. The CMMI for Development contains 22 process areas, while the CMMI for Services has 24.

Organisations must choose whether to use a "Staged" or "Continuous" representation. The former follows a pre-defined pattern of process areas that are defined by Maturity Level, while the latters processes are based on their interest in improving only specific areas.

Process Areas are further broken down to specify Specific Goals (SGs) and Specific Practices (SPs). These practices define the expected behaviours of projects and organizations. There are also twelve Generic Practices (GPs) that provide guidance for organizational excellence including behaviours such as setting expectations, training, measuring quality, monitoring process performance, and evaluating compliance. [5]

Organisations are then rated based on the designing of these behaviours. However, these ratings are broadly defined and broadly interpreted, explaining why CMMI ratings are not a standard, as opposed to the International Organization for Standardization (ISO) requirements. There are five Maturity ratings in the CMMI:

Level 1: Initial - Process is poorly controlled and unpredictable.
Level 2: Managed - Process has met all generic goals.
Level 3: Defined - Process meets all specific and generic goals and is proactive.
Level 4: Quantitatively Managed -All goals are met and process performance newline data are collected and statistically analysed and impacts future decisions
Level 5: Optimizing - All goals are met and the focus is now on continuous process improvement

The Standard CMMI Appraisal Method for Process Improvement (SCAMPI) is a computational method that is tasked to a Certified SCAMPI Lead Appraiser to help your organisation reach a specific level or fine tune its processes. The appraisal process is divided into three categories depending on the organisation's needs. SCAMPI A is the main appraisal method and is the only appraisal method that can result in a Maturity Level Rating. SCAMPI B is used to identify user views of a product and SCAMPI C is typically used as a gap analysis and data collection tool. SCAMPI A results are published on the CMMI Institute Website known as PARS making the results available for public viewing. [5]

The aforementioned International Organization for Standardization (ISO) can also be used to compute or assess data. ISO 9000, specifically SO/IEC 90003 Software engineering, details standards to organisations the acquisition, supply, development, operation and maintenance of computer software and related support services. This variant of the ISO was developed by technical committee ISO/IEC JTC 1/SC 7 Software and systems engineering. ISO/IEC

9000-3 was published in December 1997, but was updated to become ISO/IEC 90003 in February 2004. ISO 90003 has a renew cycle of every 5 years. [9]

The ISO/IEC 90003:2014 adopts the ISO structure in 8 chapters in the following breakdown:

1:	Scope
2:	Normative references
3:	Terms and definitions
4:	Quality management system.
5:	Management responsibility
6:	Resource management
7:	Product realization
8:	Measurement, analysis and improvement

With the rise of organisations realising "you can't manage what you can't measure", the future of automated computational functions for collecting and assessing is bright.

5 Ethics Concerns

Given the rise of large data driven organisations such as Facebook or Google, the data collected on individual consumers is rapidly increasing in scope and volume, and this trend will undoubtedly continue as technology becomes a larger part of our lives. The ethical collection and usage of data is an important issue that businesses need to examine during this unique period in history, as there is a fine line between the use and misuse of data. Implementing a Privacy Policy is the first step towards governing ethical data collection, storage and usage in a business, however this is only a foundation. Internal privacy practices, ethical expectations for staff, and regular auditing of privacy practices are all crucial to maintaining an ethical data sytem.

The collection, assessment, storage and management of data gives rise to ethical concerns regarding the methods used in this process. Data is often separated into two categorises when discussing the ethical concerns regarding its usage. Prospective Data is information that was gathered with explicit consent. Participants must be aware of the data that is being gathered, how the data will be stored and the full extent to which the data will be analysed. Retrospective Data occurs when event or outcome that you are studying has already occurred at the time of data collection.

The first ethical concern regarding information is data storage. While participants in data collection studies should be told who has access to the data, the data must be securely stored for this to hold true. The type of data determines the implications for its storage:

Identifiable:	Data that contains names or photographical evidence.
Potentially Identifiable:	Data that could be linked to a person.
De-identified:	All sensitive information has been whitewashed.
Anonymous:	Personal information was never given.

Data that is identifiable or potentially identifiable is more sensitive than anonymous data and should be treated as such. However, consenting participants should be warned of the potential of a data leak; this consent does that transfer to Retrospective Data, giving rise to further ethical implications regarding the use of this data. Retrospective data is often created when data is archived for secondary analysis, resulting in the ability to validate or refute conclusions and reducing the burden of repetitive data collection. While data archiving provides an obvious benefit to science, ethical concerns must be raised as the publication of data could result in the data being used in ways that the participant did not willingly consent to. To address this issue, researchers must ensure the future of the data collected is clearly conveyed to the participants and a distinction between formal managed archives verses the open publication of data must be identified. [13] When data is collected and stored for the sole use of improving consumer interactions, internal business ethics must be developed and utilised throughout the business. For example, Consumer Relations Management (CRM) data should only be accessible by the relevant areas of the organisation and should only be accessed on a need to know basis.

When evaluating ethical concerns of data collection and analysis during the software engineering process, the question of explicit consent can become blurred in the goal of gathering accurate data that truly reflects the workplace environment. For example, suppose a company plans to examine the number of lines of code each employee writes on average per day. Informing each employee about the data collection, the use of the data and the length of the data collection period is likely to skew the results away from a true representation of the lines written by each employee, as employees would be more likely artificially improve their lines written per day as they now understand that this aspect of their work will be examined. In situations such as this, the ethics can become blurred as one must often decide to respect autonomy and consent in exchange for truly accurate results.

The ethical issues around data collection, storage and usage can appear confusing, making it difficult for an organisation or researcher to discern if they are acting ethically and legally. Following relevant data privacy laws is essential, as is having general respect for the customers interests. We must ensure that we only collect the data we need and have explicit consent to gather, store the data in a secure manner, and clearly disclose to participants in your Privacy Policy what we plan to do with the data, and the reason for such usage.

6 Conclusion

Since the development of software engineering in the early 1950s, the idea of how best to measure such as process has developed coincidingly. Imperial quality measurements such as speed, usability, testability, readability, size, cost, security, and number of flaws or "bugs" have provided a basis for measuring and analysing software engineering. Further measurements, although harder to quantify such conciseness, and customer satisfaction and benefit to society have also been suggested as a basis for identifying the success of the software engineering process. As the technology grows alongside this process, the collection of data and analysis of data improves through computational automation, improving the process as a whole, but giving rise to ethical concerns that must be contemplated. To conclude on a quote from published software engineer Tom Cargill, "The first 90 percent of the code accounts for the first 90 percent of the development time...The remaining 10 percent of the code accounts for the other 90 percent of the development time", but future advancements in the software engineering are promising to change this trend.

References

[1] *IEEE Standard Glossary of Software Engineering Terminology*, IEEE std 610.12-1990, 1990.

[2] M. Marmer, B. L. Herrmann, E. Dogrultan, R. Berman, C. Eesley, S. Blank, *Startup Genome Report Extra: Premature Scaling* , Startup Genome (2011).

[3] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach, Second 2nd edition revised ed.* Boston: PWS Publishing, 1997.

[4] N.E Fenton, *Software Metrics: a Practitioner's Guide to Improved Product Development* Software Engineering Journal, 9(1), 1994,

[5] What Is Capability Maturity Model Integration (CMMI)? — CMMI Institute", Cmmiinstitute.com, 2017. [Online]. Available: http://cmmiinstitute.com/capability-maturity-model-integration. [Accessed: 06- Dec- 2017].

[6] CoSEEEK, "CoSEEEK", SourceForge, 2017. [Online]. Available: https://sourceforge.net/projects/coseeek/. [Accessed: 06- Dec- 2017].

[7] "Measurement System Analysis (MSA) Tutorial", Moresteam.com, 2017. [Online]. Available: https://www.moresteam.com/toolbox/measurement-system-analysis.cfm. [Accessed: 06- Dec- 2017].

[8] "Software metrics and measurement - Wikiversity", En.wikiversity.org, 2017. [Online]. Available: https://en.wikiversity.org/wiki/Software_metrics_and_measurement. [Accessed: 06- Dec- 2017].

[9] ISO/IEC 90003:2014 - Software engineering – Guidelines for the application of ISO 9001:2008 to computer software", Iso.org, 2017. [Online]. Available: https://www.iso.org/standard/66240.html. [Accessed: 06- Dec- 2017].

[10] Fenton, N. E., and Martin, N. (1999) "Software metrics: successes, failures and new directions." Journal of Systems and Software 47.2 pp. 149-157.

[11] Stephen H. Kan. 2002. Metrics and Models in Software Quality Engineering (2nd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[12] Grambow, G., Oberhauser, R. and Reichert, M. (2013) Automated Software Engineering Process Assessment: Supporting Diverse Models using an Ontology. Int'l Journal on Advances in Software, 6 (1 & 2). pp. 213-224.

[13] A. Sillitti, A. Janes, G. Succi, and T. Vernazza, Collecting, integrating and analyzing software metrics and personal software process data, in Proceedings of the 29th Euromicro Conference. IEEE, 2003, pp. 336 342.

[14] Johnson, Philip M., and Hongbing Kou. "Automated recognition of test-driven development with Zorro." Agile Conference (AGILE), 2007. IEEE, 2007.

[15] Frederick P. Brooks, Jr, "No Silver Bullet Essence and Accidents of Software Engineering", Computer, vol. 20, no. 4, pp. 10-19, 1987.

[16] Orthogonal Defect Classification (ODC) in Agile Development. M. Jagia, S. Meena, IEEE ISSRE 2009 Supplemental Proceedings, Nov. 2009.