Software Measurement Report

Chloe Conneely 17323080

# Contents

Introduction

Measurable Data

Computational Platforms Available

Ethics Concerns

Sources

# Introduction

This report documents the measurement of the process of software engineering. In this report I will document the various types of data that are measured in the software engineering process, the computational platforms available for use by software engineers to track the data, the various algorithms designed to process and evaluate the data measured and the ethical implications of the gathering and measurement of such data.

# Measurable Data

As software engineering is a data heavy process, it is only natural that software engineers have selected numerous metrics for measuring the progress and quality of software.

Lines of code: Measuring the lines of code written as a way of tracking the productivity of a software engineer and program quality. This measurement has been used since late 1960s(4).

Defect counts: Counting the number of software defects found in software development and testing(4).

McCabe’s Cyclomatic Number: This formula measures code complexity using a graph representation, M = E – N + 2P where E: number of edges of the graph N: number of nodes P: number of connected components(6) M : complexity(5)

Function point analysis: This data “assesses the functionality delivered to its users, based on the user’s external view of the functional requirements”(7).It is popular in financial IT sector(5).

Time working on a project: Measuring the time spent by a software engineer working on files pertaining to a project in an IDE (integrated Development Environment) such as Eclipse or Emacs.

Frequency of commits: The frequency of a software engineer committing code to a code repository. This can also check for the number of lines of code committed to the repository.

Number of builds: The number of times a software engineer builds the code in order to run it. This can also check for the number of successful builds.

Times tests are called: The number of times a software engineer calls tests. This can also measure the number of successful tests in a call.

Topic-author association: Searching the source code for keywords or topics and associating authors with the topics based on the history in the software repository(10). This can be used to identify an author familiar with a topic used in the code along with subject matter experts in areas of the code. This cannot be used to determine the core developers of the project and does not take advantage of relationships between authors found in repository(10).

Code Coverage: The percentage of code that is tested by a software engineer in the unit test stage of software development.

Keystrokes per Second: The average number of keystrokes a software engineer in pressing per second.

Comments: This can be measured as the number of comments included in the code or the percentage of the code which are comments.

Classes: The number of classes created by a software engineer in a project.

Code churn: The number of lines of code that were added, changed or deleted in a given time. The information given in relation to it is: User, Date/Time, Project, Path(17).

Application crash rate (ACR): Calculating the number of times an application fails.

ACR = F/U

F: number of fails

U: times application is used(17).

Assignment scope: The number of projects taken on by a software engineer in a year(17).

# Computational Platforms Available

In this section I will describe some of the platforms available to analyse the data that is measured in software engineering. These are:

* PSP
* Leap
* Hackystat
* CMMI
* SPICE
* SDSA
* Zorro
* Agile

The use of social technologies is prevalent in the IT industry as a whole with 57% of companies using them.(2)

PSP: Personal Software Process (PSP) is a measurement platform that was described in Watts Humphrey’s *A Discipline for Software Engineering*.(1) This book “showed how to adapt organisational software process analytics for individual developers[,] … how these analytics could drive improvement [and] … presented the practices in an incremental fashion amenable to academic and professional adoption”.(1) PSP’s primary goals are “to improve project estimation and quality assurance”(11).In PSP developers must fill out forms describing a: project plan summary, time reading log, defect recording log, process improvement proposal, size estimation template, design checklist and a code checklist.(1) The forms contain over 500 unique values that the developer must manually calculate. Developers can modify it by choosing the analytics that best suit their personal requirements. When PSP is taught to students it is quickly abandoned due to the “extra work” involved in using it(11).

Leap: **L**ightweight **E**mpirical, **A**ntimeasurement dysfunction and **P**ortable software process measurement (Leap) toolkit automates and normalises the data analysis of PSP. The developer manually enters most data and “the toolkit automates subsequent PSP analyses and in some places provides analyses … that the PSP doesn’t provide”.(1)  It only maintains data about the developers activities and doesn’t reference developer’s names in data files(1) It also has a repository of personal process data for developers to keep as the work of various projects. A developer is required to design and implement a toolkit component if they want to analyse a condition/variable that is not premade in the toolkit.

Hackystat: Hackystat implements “sensors attached to development tools [which] gather process and product data and send it to a server, which other services can query to build higher-level analyses”.(1) Hackystat has four main features:

* Client- and server-side collection which are integrated into editors, build tools, test tools along with configuration management repositories, build servers and others on the server-side.
* Unobtrusive data collection. Developers don’t notice their data is being collected. Hackystat can even locally cache data that is collected while a developer works offline and then sends the data to the data repository when the developer is back online.
* Fine-grained data collection collects data on a minute-by-minute basis. There’s also buffer transition which collects data “each time the developer changes the active buffer from one file to another”(1) along with tracking when a developer changes a function, contructs a test case or calls a test.
* Personal and group based development: Hackystat tracks the interplay among developers when the edit the same file.

Hackystat also measures the time each developer spends working on the project, measures the frequency of commits, number of builds, number of times the tests are called. Access to the data collected by Hackystat requires a password that should be known only to the developer in question(11).

Capability Maturity Model Integration (CMMI) is comprised of five maturity levels: 1-Initial, 2-Managed, 3- Defined, 4-Quantitatively Managed, 5-Optimising(8). These levels describe the “degree of process improvement across a predefined set of process areas, in which all goals within the set are attained”(9). Each maturity level has process categories which have process areas containing specific goals with specific practices. Assessment is quantified by a performance scale: 1-unrated, 2-not applicable, 3-unsatisfied, 4-satisfied. The process assessment is applied by rating each generic and specific goal of a process area using the performance scale. A maturity is achieved if all process areas within the level and within each lower level are either 2 or 4(8). CMMI also features generic goals with generic practices with the capability levels: 0-Incomplete, 1-Performed, 2-Managed, 3-Defined, 4-Quantitatively Managed, 5-Optimising. These indicate “Achievement of process improvement within a process area”(9).

Software Process Improvement and Capability Determination (SPICE): SPICE is comprised of six capability levels: 0-Incomplete, 1-Performed Process, 2-Managed Process, 3-Established Process, 4-Predictable Process, 5-Optimising Process. Each level has one or more process attibutes(8). SPICE assesses performance through the following scale: 1-Not Achieved (0%-15%), 2-Partially Achieved (16%-50%), 3-Largely Achieved (51%-85%), 4-Fully Achieved (86%-100%). The assessments are used to assign to each process one or more capability levels and use them to calculate assessments for projects rather than determine if a capability level is achieved.

Software Development System Analysis (SDSA): SDSA is a Hackystat application providing a generic framework for organising and analysing the data Hackystat receives as the input to a rule-based time-series analysis. SDSA merges the data collected into a single sequence which is ordered by timestamp, called the “development stream”(14). Next the sequences are tokenised which results in a sequence of episodes.

Zorro: Designed to be an extension of Hackystat and SDSA it gathers developer behaviours, organises them into episodes and applies a rule-based system to determine if an episode is an instance of test driven development(14). From Hackystat, Zorro collects: unit test invocations, compilation events, refactoring events and editing events. Zorro extends SDSA by classifies the episodes into one of 22 types. The 22 types are arranged into 8 categories: Test First (TF), Refactoring (RF), Test Last (TL), Test Addition (TA), Regression (RG), Code Production (CP), Long (LN) and Unknown (UN). The final step is for Zorro to determine the test driven development conformance of the instance.

Agile: In 2001 the “Agile Manifesto” was published(15). The aims were: “uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value: • Individuals and interaction over process and tools, • Working software over comprehensive documentation, • Customer collaboration over contract negotiation, • Responding to change over following a plan. That is, while there is a value in the items on the right, we value the items on the left more.”(16)  In Agile user requirements are defined as stories that are gathered in a backlog. Effort estimation is subjective with the team collectively assigning User Story Points (USP). Velocity is the number of completed user stories and is often used to estimate the remaining time to end a project(15). Velocity is a measure for productivity and is represented as a graph. A burndown chart represents the amount of work remaining. It compares estimated work (ideal burndown) with remained work which helps the team in deciding on adding or dropping stories based on the project’s link to time remaining. Cumulative Flow Diagram (CFD) “presents a quantity of work in a given state”(15). Can be used to increase throughput, reduce lead-time, correct problems early and monitor/eliminate bottlenecks. Re-work allows for flexibility by defining the man-hours spent fixing defects(15). A re-work graph is useful for discovering bottlenecks and delays. Earned Business Value (EBV): EBV tracks the tracks the value of the software. EBV is calculated by breaking down the project “based on the extracted and refined features and user stories”(15). A formula for calculating EBV is defined as the sum of weight for stories done. EBV can only be calculated when the scope of the project is clearly defined. Project duration is estimated based on the number of full-time team members.

# Algorithmic Approaches Available

In 2015 Hitachi published a study on “Measuring Happiness Using Wearable Technology”.(3) Hitachi created a device for employees to wear like an employee badge is worn so that they could try to quantify happiness. They developed the 1/T rule (T meaning the duration of activity) which “categorizes physical activity during each unit of time as either inactive or active, and looks at the ‘active’ times when the person is moving and how long they last (called ‘sustained activity’). Activities categorized as ‘active’ include not only walking, but also small movements such as nodding or typing.” (3).Their research revealed that the distribution of T values in 1/T fluctuation (the changes in the rate of physical activity) in a group took a long-tail distribution. They noticed that “some groups follow the rule closely while others diverge considerably.”(3) To determine if these statistics show a correlation between activity and happiness, the researchers asked a questionnaire designed to determine depression. The questionnaire contains 20 questions and each question asks for a response between 0 to 3 in relation to happiness, the answer value is then multiplied by the number of questions to which it relates (for example, a person giving the answer 3 for each question will have a result of 3\*20 =60 indicating a high level of happiness).(3) The researchers found a high correlation between 1/T fluctuation and departmental mean values of happiness from the participants, noting that groups with a “high level of happiness also have a high level of diversity in their body movements” (3). They also notice this correlation resulted in improved job performance. The researchers concluded that it is possible to use AI in the workplace to make the work environment a place that increases the happiness of the workers through the wearing of the badge-like device.

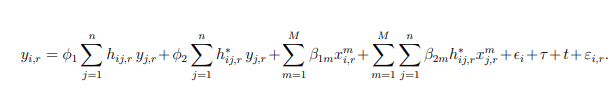
Context-aware Software Engineering Environment Event-driven framework (CoSEEEK) is designed to “improve [the software engineering] process support and guidance in an automated fashion”(8). CoSEEEK’s functionality includes:

* Quality Management: improves automated detection of quality issues in source code by facilitating the automated assignment of software quality measures to counteract such issues(8)
* Knowledge Management: enables collection and management of knowledge in projects by semantic enhancement so CoSEEEK can automatically provide appropriate knowledge to users at appropriate times(8)
* Exception Handling: exception handling, the person doing the handling and the appropriate time for handling can be automatically determined(8)
* Task Coordination: automatically coordinates activities of different areas of a project through automatic notification in case if certain changes of artifacts/activities and automatic issuing of follow-up actions required by other actions(8)
* Extended Process Support: facilities to implement a larger coverage of activities in projects as process models. Can model dynamic activities using a model and transformation facility(8)

The algorithm used by CoSEEEK in assessment requires a concrete project and Assessment Process Template used for the project. The first step of the algorithm creates a structure comprised of Process Categories, Processes and Base Practices for the Assessment Process containing functions such as createConcept which creates a concept from a given template and linkConcepts which links two concepts together. In the next step, the algorithm creates the Capability Level structure which creates Capability Levels and their Determinators. Then the Determinators are linked to the Base Practices used for determining capability using the function getRelatedBasePractices. This function gets all Bases Practices in the Assessment Process that are to be connected to a certain Capability Determinator. For each Base Practice, a Rating is created linking them to the Capability Determinator. A standard Perfomance Scale and a responsible person are attached to each rating. The final step relates to assessment. Throughout the project an automated rating is applied when matching events or status changes occur. A manual rating is applied at the end of a project. This distributes the rating information to the responsible person. The person can check the automated rating, rate practices that have not been rated or distribute parts of the assessment to others who can provide missing information for rating practices. This process is concluded by checking the achievement for each Capability Level of an Assessment Process (8).

There are algorithms that exist to identify communities of authors in projects based off data from repositories such as iGraph’s \_fastgreedy method and the spinglass method. \_fastgreedy runs in O(n log2 n) time(10). It breaks a network into communities in a way that maximises the total modularity of the network. It can be useful for identifying small teams of programmers to work together(10). The spinglass method associates negative modularity with the energy of an infinite range spin glass and attempts to minimise the energy of the system to find communities(10). This algorithm can be used to set up the best team structure by choosing people that share a collaboration history.

A study that was published in 2015 used algorithms to examine the productivity of employees based on peer pressure through working in a network(12). The first algorithm focused on network effects on worker productivity:

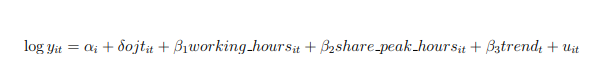


φ1 and φ2 represent the local aggregate and the local average network effects (respectively)

εi,r represents i.i.d. innovations with zero mean and variance σ 2 for all i and r

The characteristics x m i,r and x m j,r are gender, age, tenure, total work hours during the week, day(s) of the week worked and the time of the day worked (morning, midday, evening)(12)

The second algorithm focused on network effects of worker productivity from on the job training:



yit is the agent’s performance in week t

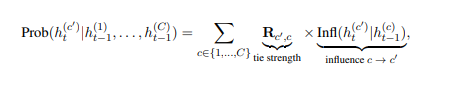
ojtit is and indicator variable for on-the-job training

Treatment status is randomized by team so that δ represents the average treatment effect

Baseline regression also includes individual fixed effects, αi , the number of hours worked during a particular week, working\_hoursit, whether or not a worker’s shift coincides with the peak workload hours, share\_peak\_hoursit, and a linear time time trend, trendt (12)

The study concluded that peer pressure improves productivity as it revealed that a “10% increase in the current productivity of a worker’s co-worker network leads to a 1.7% increase in their own productivity”(12). This study also was able to use the algorithms to determine the strongest and weakest employees.

A study published in 2012 investigated the ability of modelling dynamic influence in human interaction. The model contains C entities, with each C having 1-S states. Time t each entity is in one of the states ht (c) ∈ {1, . . . , S}. The report assumes each entity’s latent state is constant(13). The influence model is:



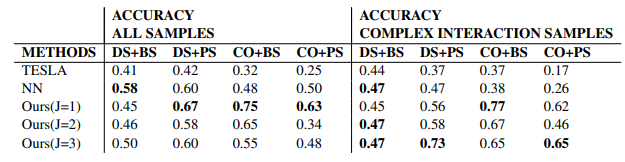
R is a C X C matrix

Rc1,c2 represents the element at the row c1 column c2 of matrix R

Each row in the matrix is stochatic i.e sums up to one(13)

 is modelled using an S X S row stochastic matrix Mc,c’ allowing  where  represents the element at row  and column of Mc,c’.

The researchers tested the influence model by attempting to predict turn taking in a discussion. Their results are shown below:

(13)

# Ethics Concerns

I find the gathering of data on a broad scale to be extremely concerning especially regarding privacy and employee autonomy.

I understand that the measurement of software engineering practices is a useful aid for software engineering to be carried out in a way that achieves the best results. For example, in my source 14, the developers of Hackystat encourage individuals from all over the world to test Hackystat as they are eager to improve the software from the perspective of analysing the data produced so they can research the process of software engineering(14). Hackystat is an opensource application so it is easily available on the internet to anyone who wishes to use it. Perhaps I am naïve but I am less concerned about data that is gathered for academic/research purposes as I imagine that data isn’t going to be used for nefarious purposes.

However, due to Hackystat being opensource, there are companies that use it to measure the software engineering process of their employees(1). One of the reasons companies use software like Hackystat is to determine the strongest and weakest employees. While this is an understandable desire for companies, my concern here is that the companies won’t explore why their weakest employees are the weakest or if the weakest employees are affecting the overall quality of work done in the company (i.e. are the weakest employees causing the company to miss deadlines). This could result in companies firing employees over matters that could be easily resolved through training or learning why these employees are the weakest. Also, by using data collecting software to track the progress of employees, this can increase stress and anxiety among employees since they will feel an expectation to always perform at the standard the company has set which may or may not be realistic or healthy. In source 1 the authors of the report not that the employees that use Hackystat at work feel uncomfortable with the degree of data collection done by Hackystat, especially since it collects the data without informing them(1).

Continuing with the topic of the wellbeing of employees, another concern with the broad collection of data produced by software engineers is employees comparing their performance with co-workers that could have better results based on the data produced. The comparison of employees based on the data measured in these platforms could make employees feel insecure about their abilities and such insecurity can negatively impact their job performance. This should not be a concern if the employees are in a positive work environment.

Privacy concerns for employees also extend to what the employees do during their time in their place of work. In source 3, the researchers at Hitachi discuss their use of a badge that employees wear in order for it to measure their happiness(3). While this sounds pleasant at first, an employer concerned about the mental health of their employees, the badge raises concerns about personal privacy. The badge must be worn by the employees for the whole workday, including at breaktimes. The badge is used to measure the level of physical activity done by the employees as the premise of the research was based on the link between physical activity and happiness(3). The report mentions the possibility of AI being used in the badge to calculate exactly what must be done in order for employees to be happy in their workplace. My concern here is that a company possessing such large and varying amounts of data can use it to manipulate the lives of their employees in a way that will benefit the company but not the employees, giving an employer total control over their employees.

Another privacy concern is the security of the information gathered in the software engineering process. I know that any data that is gathered by a person or organisation will not be safe from an unauthorised individual or group accessing it. This is a significant problem if the data accessed is sensitive information such as passwords for services or bank account information. Source 2 notes that 55% of company leaders across all industries are concerned that the use of social technologies increases the likelihood of confidential information being leaked(2). Even if the data isn’t confidential, how this data is stored /managed is of great concern to me. The practice of companies selling customer information to third-parties is widely known and I personally find it to be disgraceful because I think it is another way for already profitable companies to make more money at the expense of their employees and customers. Source 2 mentions one third of companies analyse customers social data and only 19% report using customer data in predictive analytics(2). Again, I am not enthusiastic about companies hoarding data because the purposes are not exactly noble or honest.

The widespread gathering of data is also used in the development of AI. With this comes the same concerns about privacy I discussed in the previous paragraphs. As researchers spend more time developing AI technology the quality of the technology is ever increasing. In source 3, Hitachi suggest that AI can be used to control the temperature of an office to make the office more comfortable for the employees based off the measurements from the computerised badges(3). Source 13 describes an experiment that tests the ability of AI to predict when a person is going to speak first in a group. The results were encouraging to the researchers as there were situation where the badge was 50%-70% accurate in its predictions in contrast to human estimation being 50% accurate and random estimation being 33% accurate(13).

While the developments in AI are exciting to researchers, there is the concern that AI could be used to replicate the behaviour of humans. Google created a bot that is “barely distinguishable from a person”(18). The bot can make appointments for restaurants. The bot even imitates the pauses and terms people use when thinking such as “um” and “ah”, giving it a very lifelike voice. My concern with this kind of technology is that it could be used to further automate jobs that involve customer interaction such as call centres and tills. Automation is useful for a company as they save money on wages but in places like a call centre, automation makes seeking contact with the company unproductive as the current technology for AI is not at the level where it can handle the variety of issues that can arise in customer service. Even with the Google bot’s sophistication, I still think there’s a long way to go before AI will be able to handle unanticipated interactions with humans.

To return to the concerns of privacy, I will include a concern found in AI. The rise of facial recognition software has led some, including myself, to be concerned about the potential of AI being used for mass surveillance(19). The lack of regulation due to the rapid development of AI has resulted in its use for monitoring the activities of people. In China, AI is being used to try and predict if a person will commit a crime. This involves using facial recognition and “’personal re-identification’ — matching someone’s identity even if spotted in different places wearing different clothes”(20). I think using AI to guess if a person is going to do something is a very reckless and an overreaction to the fear that people have autonomy, meaning no organisation or entity can supress human behaviour, even with technological advances.

My ethics concerns lead me to the conclusion that actions that involve data collection should have legal regulations in place to protect the privacy of individuals and prevent any information gathered to be used in a malicious manner.

# Sources

1. <http://www.citeulike.org/group/3370/article/12458067>
2. http://www.nextlearning.nl/wp-content/uploads/sites/11/2015/02/McKinsey-on-Impact-social-technologies.pdf
3. <http://www.hitachi.com/rev/pdf/2015/r2015_08_116.pdf>
4. Fenton, N. E., and Martin, N. (1999) "Software metrics: successes, failures and new directions." Journal of Systems and Software 47.2 pp. 149-157.
5. <https://dzone.com/articles/what-exactly-is-mccabe-cyclomatic-complexity>
6. <https://en.wikipedia.org/wiki/Cyclomatic_complexity>
7. <https://www.geeksforgeeks.org/software-engineering-functional-point-fp-analysis/>
8. 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.
9. CMMI Product Team, “CMMI for Development, Version 1.3,” Software Engineering Institute, Carnegie Mellon University, 2010
10. Dittrich, Andrew, Mehmet Hadi Gunes, and Sergiu Dascalu. (2013) "Network Analysis of Software Repositories: Identifying Subject Matter Experts." Complex Networks. Springer Berlin Heidelberg. pp. 187-198.
11. P.M. Johnson et al., “Beyond the Personal Software Process: Metrics Collection and Analysis for the Differently Disciplined,” Proc. 25th Int’l Conf. Software Eng. (ICSE 03), IEEE CS, 2003, pp. 641–646.
12. <https://www.researchgate.net/profile/Jan_Sauermann2/publication/285356496_Network_Effects_on_Worker_Productivity/links/565d91c508ae4988a7bc7397.pdf>
13. W. Pan, W. Dong, M. Cebrian, T. Kim, J. H. Fowler and A. S. Pentland, "Modeling Dynamical Influence in Human Interaction: Using data to make better inferences about influence within social systems," in IEEE Signal Processing Magazine, vol. 29, no. 2, pp. 77-86, March 2012.
14. Johnson, Philip M., and Hongbing Kou. "Automated recognition of test-driven development with Zorro." Agile Conference (AGILE), 2007. IEEE, 2007.
15. Taghi Javdani , Hazura Zulzalil, Abd. Azim Abd. Ghani, Abubakar Md. Sultan, On the current measurement practices in agile software development, International Journal of Computer Science Issues, 2012, Vol. 9, Issue 4, No. 3, pp. 127-133.
16. K. Beck, A. Cockburn, R. Jeffries, and J. Highsmith. (2001). Agile manifesto, <http://www.agilemanifesto.org>
17. <https://stackify.com/track-software-metrics/>
18. <https://www.wired.com/story/google-duplex-pixel-smartphone/>
19. <https://www.globalgovernmentforum.com/eu-report-warns-of-ai-risks-to-human-rights/>
20. <https://www.ft.com/content/5ec7093c-6e06-11e7-b9c7-15af748b60d0>