David Noonan - 15320927

# The Measurement of software engineers

Measurement of software engineers as in all work is crucial to the management process but in a field where the complexity of the work is so high a manager must find new ways of analyzing their workers. The end result of a construction job for example is easy to evaluate. There are a set of rules that must be followed and there is entire fields of jobs to ensure people are following those rules. Due to the nature of the work it is also easy to judge the pace of the worker and the quality of their work. For software engineers its quite different. There are many metrics to collect and analyze to get a full picture.

There are two categories of measurement I am going to analyze in this section physical and digital. Both have their pros and cons and a good mix of both seems to be the only solution for the time being.

The physical method of recording software engineers is quite simply asking them to provide the information. This can come in may forms such as filling out reports or progress meetings.

In Software project management By Bob Huges and Mike Cotterall they discuss this. They say there are five different forms of progress analysis which are

Oral formal regular - This is a regular meeting with the manager possibly once monthly or weekly to discuss the progress. This is a useful tool as the regularity of the meetings allow you to observe if the pace is consistent throughout the project.

Oral formal ad hoc – This would not be a regular meeting but possibly a meeting after a stage of development has been completed. This allows the manager to be up to date with the development and get information to provide to their higher ups.

Written formal regular – Examples of these are job sheets or progress reports. These tools are useful as they can be quick if done right and are easily referenced.

Written formal ad hoc – Much like the regular the written element of these is quite convenient. Examples of these are change reports.

Both of the oral methods require two participants to be available at the same time and minutes will be required for a record of the meeting. As in all meeting you also need a meeting space. For the written methods the disadvantage is it may be hard to get the worker to participate regularly due to forgetfulness or possibly the sheer monotony of filling out the reports. This can cause time wasted following up on these forms.

The last method proposed is Oral Informal Ad hoc this is a casual conversation with the worker at the water fountain or lunch table. This is a difficult method to employ as you much have quite a good relationship with the person to have it be effective. It is however quite a good method as it can show early warnings much easier than any other method.

This must of course be followed up with an official report however which might not be received well by the worker.

In Searching under the streetlight for Useful Software Analytics Philip M. Johnson provides a list of forms for developers to fill out to provide a more in depth look into their work for analytic uses.

These include

* A project plan summary
* A time-recording log
* A defect recording log
* A process improvement proposal
* A size estimation template
* A time estimation template
* A design checklist
* A code checklist

In addition, software engineers are workers and all the usual recording methods will be useful such as clock in times and days absent. While obvious it would be a mistake to ignore them.

The physical method of collection has its drawbacks in the time required to collect the data and the accuracy of the data. The time component is small per metric collected but to be effective you must collect a wide assortment of data to have the ability to cross examine and find the productivity lapses. This can be quite frustrating for the engineers also as it can disrupt their workflow and cause unnecessary stress.

The second aspect measurement I’m going to examine is the automatic measurement of engineers. This involves installing programs on the computers of the engineers which will track certain aspects of their work. This is also being expanded on in recent years to include wearable technology which can allow for the collection of much more far reaching metrics.

The types of data to collect is a topic of much discussion in the world of software engineering. Each metric has its benefits but also each one has a workaround with which the engineer can trick the system into thinking they are preforming at a much higher standard than is actually the case.

The most common things to record with theses tools are simple metrics like time spend at the computer and number of key strokes. But with the new technology this can be expanded to how many steps each person is taking a day and even who each person is talking to in the office. This type of data is very useful as it can show who works well with each other and who is slacking off on a regular basis.

Software can be used to examine code after it has been written to examine it on its usefulness and efficiency. Code complexity can be examined in this way along with integrity.

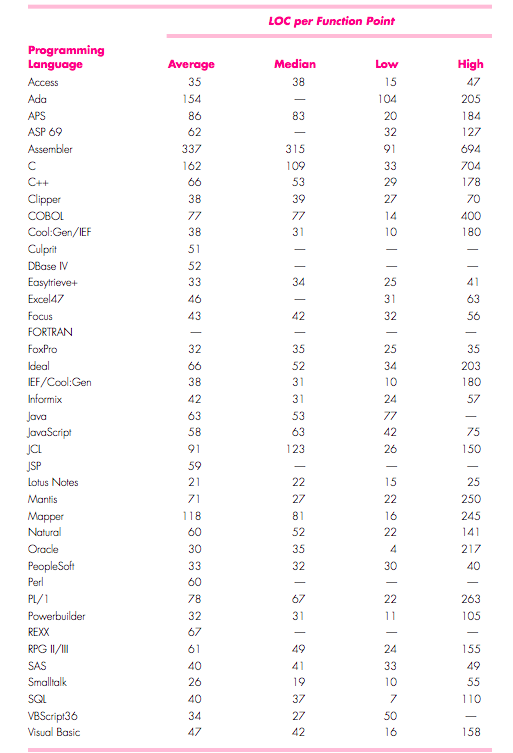
These are difficult and costly to measure if the process is not automated.

Another flaw of the automated method is that if used exclusively it has no outlet for employees to voice their opinions on how to improve their own productivity.

In summary both methods have their positives and negatives and as with most things a healthy balance must be stricken to be effective and efficient. There will always be a place for direct observance and reporting of progress but the future of the industry is to collect as much data as humanly possible on the engineers and to have that at your disposal.

In this section I am going to look at a few of the most common metrics that can be collected and their pros and cons. First I am going to look at size orientated metrics to judge progress.

The first metric I am going to look at is lines of code. This metric seems like the most obvious solution. it is so easy to collect it is even displayed on the side of every IDE. However, as all software engineers should know this is the one of the most useless metrics available. Any person who knows how to program knows how to beat this by just adding a new line at all possible opportunities. There is also the problem that even if you weren’t doing this you would still run into the problem that the more code you write to solve a problem the less efficient it is. For these reasons it is a terrible idea to measure programmers on how much code they can write in the words of Bill Gates "Measuring programming progress by lines of code is like measuring aircraft building progress by weight.", however there is some merit in using lines of code in conjunction with an other metric. The term KLOC or thousand lines of code can be used to measure errors or documentation. Lines of code is also language dependent so many times it cannot be used across projects. data and provide analytics for the managers this of course would incur an additional cost to the endeavour.

The next metric I am going to examine is function point analysis. It is the successor of lines of code as it solves some of the major problems it suffers from. Function point analysis uses actual end user functionality as a measurement. Gaining one point per actual function you can provide. This solve the code inflation problem as it gives no incentive to add extra lines to code. In addition, function points are consistent across all languages allowing for easier comparisons as shown in the table below. In the table it shows that C++ has 2.4 time the functionality as one line of code of C.

The nature of function points makes it much easier to do estimations from the beginning of a project thus giving more accurate progress appraisals. Function points are however highly correlated to lines of code meaning despite all the improvements they still don’t provide much more useable information.

Both lines of code and function points are relatively accurate estimators of development progress however they have very little use until a sizeable amount of data has been collected (most likely form previous projects).

The next area I will examine is code quality. This is a much more useful measure but also much harder to collect data on.

The first measure I will look at is the most basic. Correctness is a measure of how well a piece of software does its job. It is commonly measured through defects per thousand lines of code. A defect is simply code that is unfit for its purpose. This is commonly measured after the release of the software through end users submitting bug reports. Which is disadvantageous as its cant be measured during production. Its is usually counted of a standard period of time such as a year which also means that you cannot obtain frequent information.

The next metric I will examine is maintainability. As most software is not written from scratch maintainability is vital to the success of an enterprise. Most code written is done so during maintenance If code is unmaintainable it will become outdated and incur huge costs the repair or replace. Maintainability is defined as the ease with which a program can be corrected if an error is encountered, adapted if its environment changes, or enhanced if the customer desires a change in requirements.

One of the metrics used to measure maintainability is mean-time-to-change this is the time it takes to fix or change part of the code. The lower the time is the more maintainable the code.

Integrity is a very important metric and has become more so recently with the increasing security breaches in even the largest companies in the world. It measures the ability of a system to resist attacks both accidental and deliberate. Integrity relies upon two variables security and threat. Security is the probability that the attack of a specific type will be repelled. It can be derived through estimation or through empirical evidence. Threat is the probability that a certain type of attack will occur within a given time period. The formula for integrity is Sum[1- (threat\*(1- security))]

For example, if threat (the probability that an attack will occur) is 0.25 and

security (the likelihood of repelling an attack) is 0.95, the integrity of the system

is 0.99 (very high). If, on the other hand, the threat probability is 0.50 and

the likelihood of repelling an attack is only 0.25, the integrity of the system is

0.63 (unacceptably low).

The last metric I will look at is Usability this is obviously a highly regarded metric as if a program is not easy to use then it is pointless to even write it. Usability is most easily tested by simply testing the product and observing if people struggle to understand it.

In this next section I am going to look at some software that is available that will automate the measurement of the software engineers.

The first system I have chosen is PSPA. PSPA is an open source software that can be used to produce automated PSP reports, graphs and summaries of projects. It does this by recording metrics such as lines of code, defect count and defect classification. This allows PSPA to produce reports on productivity per task, yield per task, defect density per task and time estimate accuracy. It also has a feature that allows the developer to record their time on each task by clicking the task name to stop and start the timer. If the developer forgets to use this PSPA will prompt them with a message. This feature allows the developer to record any distractions that occur while they are working. The interface uses a GUI and pop-up windows for ease of use. Each developer can be added as a member of a team and thus PSPA can provide team stats. Finally PSPA can be easily integrated into a some IDE’s for ease of use. All of these features show PSPA is a versatile and comprehensive tool for measuring software engineers.

Hackystat is a tool first developed in 2003 to address the issues of the first and second generation of tools as they suffered from issues of complexity and overhead. Hackeystat has client-side sensors integrated in the development tools which are compatible with tools such as Emacs, the Ant builder system and JUnit Testing. Hackeystat provides visualisations for the data in the form of reports, graphs and summaries. It automates the collection of defects and the removal of them. Hackeystat allows you to split a project into different stages to more closely analyze your data. It can provide object orientated focused recording also for when it is necessary. Hackeystat features a GUI as the main interface and also embeds other features not intended to act as user interfaces within the system such as Daily Diary which is used to visualize and explain the Hackeystat representation of a developers behavior.

Code climate is a new age company that measures your code quality and makes it easy to analyze and improve upon it. Its main use is to reduce technical debt and works optimally on a system of constant maintenance to reduce it. The interface gives you a rating on your debt which shows progress. The biggest advantage of this tool is the optimized GUI provided. This allows quick and easy access to your information and the availability of the metric is a great motivating tool for the workforce showing progress and increasing morale.

Github is possibly the most widespread tool used in software engineering. Github is a web-based Git repository hosting service and the largest of its kind in the world. Its advantages are that almost every developer is familiar with it and its interface which is intuitive and easy to use. Github tracks all commits to a project and displays previous versions. This is a very helpful tool for managing a team as it gives a visualization of all the contributors. Github also provides the service of hosting all of the code that you write in a safe repository that is accessible from anywhere. Github has tool such as milestones, project boards and track lists which are invaluable for judging the progress of a project.

In conjunction with Github there is Gitprime a piece of software that uses your Github repository or another Git service such as GitLab or BitBucket. GitPrime produces reports on the data such as daily progress and productivity. It is also useful for managing technical debt. Git prime easily integrates into an existing Github account which allows for easy adoption.

Companies around the world have begun to use wearable tech to track employees in recent years. This is not limited to software engineers but is definitely applicable to the field. An example of these is the Fitbit a very popular activity monitor. These devices allow employees to visualize their activity and therefore increase their productivity. The more active the employee is the more capable they are to work. The Fitbit provides information on sleep also with when optimized can greatly increase a persons work output. The Fitbit will allow you to monitor your heart rate and see when stress is taking its toll on your work. The gps feature will allow analysis of the worker’s movement patterns and possibly to track cross department communication. The Fitbit also has the benefit of increasing workplace wellness which of course leads to a rise in employee productivity. Due to the popularity of the fitbit companies find it easy to introduce as it is a fashionable device and the employees receive a discount on it.

The last tool I am going to look at is a device made by a company called Humanyze. Developed in the MIT media lab the company is breaking new ground in the field of monitoring and recording workers in a workplace environment. The badge is a credit card sized device that is worn around the neck of the employee that tracks communication, location, workload and activity. They do this using two microphones radio-frequency identification near field communication and Bluetooth. Using all these tools the badge can see you you are talking to and the tone of the conversation. This is extremely useful information for team leaders who need to analyze the communication between their workers to ensure that the team is working well together and that no one is being left in the dark. The Bluetooth in the card can detect who is near the user and combined with the microphone this can show the conversation between two people and the length of the interaction. The radio frequency communication allows tracking of movement within the workspace to see the high traffic areas and the activity of the workers. This device has far reaching uses and has been implemented across all ranges of workplaces including retail and banking. The founder of Humanyze is quoted as saying that by combining of digital data and real word data the Humanzye dashboard can “provide a holistic view of what goes on in a company”.

In summary there are a wide range of tools out there today to analyze software engineers. They cover a broad range of metrics and no one alone would be entirely sufficient to assess the progress and efficiency of an individual or a team. A combination of tracking code quality and the speed at which its wrote is the obvious route to follow but to look further afield at the less popular metric such as team communication and daily activity can produce results that might be more fruitful. Especially so if they have never been tried before.

The automated collection of data is quite a controversial topic. There are some ethical issues with the recording of certain data. “Unfortunately, the easier an analytic is to collect and the less controversial it is to use, the more limited its usefulness and generality” as said by Philip M Johnson in his paper Searching under the Streetlight for Useful Software Analytics.

In the age we live in most of our activity has some form of record but especially so online. It has a big business to simply sell the data a person unknowingly creates to advertisers. Companies such as Facebook and google rely almost solely on this business and much of the public has no qualms with it. None of this data is sold on an individual basis however. A vast majority of this data being used broadly to target ads at consumers. Also anyone can at anytime stop this collection of their data by simply not using the service that they provide free of charge. These facts take the edge off of this information farming but there is still a vocal component who is against it.

In business however it is a different story. The recording of the employee while extremely beneficial to the employer can also be uncomfortable to the employee. The fact that your actions can have effect beyond the targeting of advertisements and can even impact your career is worrying. Some of the examples given above such as the Humanzye badge and the fitbit collect data that could very easily reflect badly on most people if recorded unknowingly. For example, someone who goes to get water too often or talks to their cubicle neighbour too often might face repercussions even if they hit their deadlines. I think that most people when monitored at that level are doing something that would reflect badly on them. It can also be hard for a person who is detached from the work floor to see how it can effect people. Not all of the data has value and there is a price to pay to obtain it. “Innovation in the world of technology moves at a very rapid pace, but people’s understanding of how to get value from it doesn’t keep pace,” says Anthony Bruce, a partner in the UK HR consulting practice at PwC. “Right now we have solutions looking for problems; hammers looking for nails. The supply side hasn’t been clear enough about what the value proposition is.”

A state of constant vigilance by the employer seems almost dystopian to me and would lead to a very stressful situation if things got out of hand. In my opinion I think that the monitoring of an employee at such an extreme level would hurt productivity definitely in the short term and most likely in the long term. From my personal experience being under surveillance at work causes me to make more mistakes and when I do make a mistake I tend to try too hard to correct the issue and thus make it worse. While if left to my own devices (given that I am trained into the job at a decent level) I can be more relaxed, think more clearly and produce a better result. To work in one of these companies and to have my every move tracked and sound recorded would cause me to leave the job as I feel as if I can’t be trusted to work independently I don’t feel valued. Jack

“When I was consulting back in the 80s one of the things that started to show itself clearly was that the psychological contract was vulnerable to changes in technology, We have seen the darker side of that occur as often as the productivity gains.” Aiello, professor of psychology at Rutgers University, has carried out extensive research on the regulation and control of social interaction.

The persistent surveillance that is provided by these automated systems if too invasive can be detrimental to the moral of the worker. Ironically a lack of awareness would solve the adaption issues but the exposure of this would lead to far greater ethical issues. The constant recording of people can cause there to be unnecessary pressure and to reduce the effectiveness of employees (DeCaro, M. S., Thomas, R. D., Albert, N. B., & Beilock, S. L. (2011)).

Adoption of these new technologys is the main problem. Introducing such invasive methods will be fatal for the moral of the workers if done wrong. One of the solutions to this problem is to allow the engineers to choose what is recorded and what is not but that is not practical as I assume most people would avoid being recorded if at all possible. Most companies do not force their employees to take part in these programs but rather work on a system that allows them to opt out. Humanzye offers fake badges to these people so that their non compliance can go under the radar. For team management metrics this is not really an option as if one member of the team opts out then it compromises all data collected. Another is to have an independent company to interpret the data and provide analytics for the managers this of course would incur an additional cost to the endeavour. Also this means that an external company has access to this data which they can then sell on. Humanzye is an example of a company that does this. One option is to provide the data only to the engineers themselves. This would keep all data safe however without an external examiner the info might be ignored as it might be too much work to improve without motivation form a superior. The last option is to incentivise the employee some companies go as far as offering vacation days to the participants or extra pay. Sometimes it’s as simple as 50% off the new fitbit watch.

Humanzye claim that 90 percent of people opt in to their program so it is likely that people do not mind being monitored however I think that people opt in so as not to look bad in front of their employers. According to AXA’s Health Tech & You State of the Nation report, 57% of working adults would be open to wearing a fitness band or similar if supplied by their employer, and 58% would be comfortable sharing data with their employer if it was used to help improve health and wellbeing benefits.

This is an extract from the data protection act (2003)

a) by the substitution of the following subsection for subsection (1):

‘‘(1) A data controller shall, as respects personal data kept by him or her, comply with the following provisions:

(a) the data or, as the case may be, the information constituting the data shall have been obtained, and the data shall be processed, fairly,

S.2

Amendment of section 2 (collection, processing, keeping, use and disclosure of personal data) of Principal Act.

(b) the data shall be accurate and complete and, where necessary, kept up to date,

(c) the data—

(i)shall have been obtained only for one or more specified, explicit and legitimate purposes,

(ii)shall not be further processed in a manner incompatible with that purpose or those purposes,

(iii)shall be adequate, relevant and not excessive in relation to the purpose or purposes for which they were collected or are further processed, and

(iv)shall not be kept for longer than is neces- sary for that purpose or those purposes,

(d) appropriate security measures shall be taken against unauthorised access to, or unauthor- ised alteration, disclosure or destruction of, the data, in particular where the processing involves the transmission of data over a net- work, and against all other unlawful forms of processing.’’,

(b) in subsection (5), by the substitution of the following para- graph for paragraph (a):

‘‘(a) Subparagraphs (ii) and (iv) of paragraph (c) of the said subsection (1) do not apply to personal data kept for statistical or research or other scientific purposes, and the keeping of which complies with such requirements (if any) as may be prescribed for the purpose of safeguarding the fundamental rights and freedoms of data subjects, and’’,

(c) by the deletion of subsection (6), and

(d) by the substitution of the following subsections for subsec- tion (7):

from my understanding of the underlined portion of the above document the data collected in research or other scientific purposes can be stored and reused at will and the data collected by employers can be definitely be argued as scientific. This means that anything that is recorded can be kept even if you leave the company. Which to me is quite worrying.

In conclusion it is quite an interesting topic. Id expect it to not be much of an issue starting these programs due to the fact that most people do not mind being recorded. Some however will and there in lies the only downside to the endeavor. With all of this recording there is however a responsibility on the employer to be ethical with the data both in its use and its protection.

The analysis of the data produced by these method is a job for computer systems. Algorithms have been designed to use this data and produce reports that developers can use to easily visualize the data.

When humans make decisions they use a combination of past experience and intuition. This applies to all situations we experience. Computers think differently. They have to rely on hard data. When Kasparov was beaten by Deep Blue in 1996 it was done by pure brute force. The computer had been programmed in with millions upon millions of moves and it simply searched through them to find the best one. This is where computers are much better than people.

One way to utilize this to reach our goal is to get access to old repositories. Computers can go through all of this information and find useful metrics such as bug fix times and progress time.

A lot the old repositories are available online on sites like Github. The computers have the ability to learn and record these patterns which can be used in the future to prevent the errors from occurring. Some organizations are looking into this. One example is Rational insight a program for mining repository developed by IBM. Rational insight works in real time to analyze the data as its being produced and provide help during development. This is a step up form the mining of old repositories as you can be sure the data is relevant to the people working on the project however it will not have as much data available to draw from. From Github it seems that 40% of all pull requests are not merged with the original this could be evidence of widespread use of this method however there is no solid evidence.

Computers are very good at the calculation of technical debt. Technical debt comes in a few different forms. One of these is code duplication, usually expressed as a percentage with 0% as a perfect score and 100% as completely copied. The computer can search through the code and find the functions or tokens that do the same job. Reducing this will make the program more efficient and faster to write which shows the usefulness of the detection.

Another measure of technical debt is code complexity. Simple code is much easier to maintain and thus more valuable. Computers can analyze code for complexity and give a rating on its maintainability. One of the most common methods of doing this is through cyclomatic complexity. This system gives points for each complex element (if, case, switch statements) of a code unit (method, function, class etc.). You can then use this score to judge the complexity of the code unit. This system does vary from language to language but is a very useful tool. Below is an example of the system in use.

**public** **String**getAgeType( **int** age )  
{  
**if**( age < 0 )  
**throw new** **Exception** (“Invalid age”);  
**if**( age > 18 ) {  
**return**“Adult”;  
} **else {  
return**“Sorry you are not allowed to see the movie”;  
}  
}

There are three conditional decisions, two return statements, and one exception. So, the total cyclomatic complexity is six.

The next component of technical debt is test coverage. This can be broken down into two different branches, line coverage and branch coverage. Line coverage is the number of lines that have been used in at least one test case. Branch coverage is the amount of conditional branches that have been hit by test cases.

Dependency Cycles and coupling is a measure of the level of architectural quality and design of a system. A redundant cycle is dangerous as it can take down a system. The easiest way to explain this is by the following example: Assume that File A is dependent on File B, File B is dependent on File C, and File C is dependent back on File A. This example shows the problem with these cycles for instance if you wanted to change an aspect of file A you risk damaging each of its dependent files and possibly breaking the entire program. Coupling describes the number of external files that depend on a particular file. The more dependencies a file has, the more likely it is to break when something is changed on those files.

Documentation is vital for maintaining code and can sometimes be left out of a calculation of technical debt. A good way to measure this is to find out how many methods are documented and how many are not and use the ratio.

To calculate the total technical debt you need to express the total time to fix each of the problems and add it all up. Then you can then express technical debt in terms of man hours.

This can then allow you to express it in costs as they are directly related.

In conclusion possessing large amounts of data such as in this case is almost always a job for a computer. The al

* <http://ptgmedia.pearsoncmg.com/images/9780132582209/samplepages/0132582201.pdf>
* 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.
* Searching under the Streetlight for Useful Software Analytics Philip M. Johnson, University of Hawaii at Manoa
* Software project management Bob Huges and mike Cotterall
* DeCaro, M. S., Thomas, R. D., Albert, N. B., & Beilock, S. L. (2011). Choking under pressure: Multiple routes to skill failure. *Journal of Experimental Psychology: General, 140*(3), 390-406. [http://dx.doi.org/10.1037/a0023466](http://psycnet.apa.org/doi/10.1037/a0023466)

# Software Design and Development: Concepts, Methodologies, Tools, and Applications: concepts Methodologies, Tools, and Applications – ig global

* <http://www.hrmagazine.co.uk/article-details/the-ethics-of-gathering-employee-data>
* <https://www.dataprotection.ie/docs/Data-Protection-in-the-Workplace/1239.htm#10>
* <https://dataprotection.ie/viewdoc.asp?DocID=1467&ad=1#14>.
* <https://www.nitrd.gov/nitrdgroups/images/f/f1/Software_Intelligence_The_Future_of_Mining_Software_Engineering_Data_p161.pdf>
* <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=34526A0EC06E6389563EBB5EFC167AC3?doi=10.1.1.728.3689&rep=rep1&type=pdf>
* http://thinkapps.com/blog/development/technical-debt-calculation/