**Measuring Engineering**

Introduction

There is no perfect way to measure a developer’s performance. Throughout the years there have been many different techniques used to analyse a software engineer’s effort, skill and work ethic. However, there are many challenges software companies face when trying to assess performance. Firstly, there is various data which we can use to measure the work ethic of a developer and the results will differ greatly depending on the data we use. Secondly, there are ethical issues associated with these tools which can have a negative impact on a developer’s performance. What’s more, the accuracy of the data analysed and the results may not be a fair representation of a developer’s work ethic.

In this essay I will aim to highlight ways in which a software firm can assess the performance of their developers. The key components of measuring a software engineer’s skill and work ethic are as follows:

1. What data can be used and is does it give an accurate representation of performance?
2. Where can we compute such data, what firms provide this service and what software do they use?
3. What algorithms can we use?
4. Are there ethical issues regarding the data we use to measure a developer’s performance?

It must be noted that human intuition must be used when assessing a developer’s as there is no standard way to measure and associate a meaning to both code and process. Software cannot assess qualities in a developer such as communications skills and the ability to work in a team which are imperative traits in order to be a successful software engineering.

Data used to measure performance

When it comes to software engineering projects, **(2)** software companies are most interested in “human effort, quality, easiness of maintenance, cost and time”. We can use recorded data to analyse human effort but this will also increase cost and put pressure on time. **(2)** There are two types of data we can measure:

- **Product metrics** is one type of data and this refers to all physical data such as the number of commits a developer makes and how many lines of code he or she writes.

- **Process metrics** is slightly more abstract data such as effort required, production time and editing time.

Why do we measure this data?

There are 3 components as to why we measure product and process metrics. They are as follows;

1. To measure performance.
2. To measure a developers work ethic.
3. Security reasons.

Measuring performance

The most obvious reason as to why we would analyse data on a software developer is to record their performance. A software firm wants to maximise profits and to do so they must do their best to ensure all of their developers are performing adequately enough. **(2)**In order to measure performance, developers record different types of data on individual developers and also development teams. They use this data to analyse and measure the performance of a software developer or development team. Using data analytics, companies aim to ensure their developers are building projects to a high **(2)** standard and utilising their time most effectively.

**(1)** Some precautions must be taken when using recorded data to measure performance. What is important to remember is the term ‘correlation does not imply causation’. While data recorded about a software developer may find a relationship between the developer and something negative, this does not mean they are to blame. An example would be if a developer was part of two software engineering teams that failed. It could have been no fault of his that the two teams failed and perhaps he was doing a very good job and other team members were at fault. It is essential that we realise that data can give incorrect results if we don’t understand all parameters and the situation surrounding the data.

Measuring work ethic

This is slightly different to measuring performance. Work ethic refers to how much effort a developer is putting in. While quality is essential in all work a developer does, so too is being hard-working and firms can use recorded data to assess whether or not a developer has this trait. It can be hard to accurately measure work ethic and analysed data should be used alongside other methods to make an informed decision. **(5)**For example measuring the lines of code a developer writes per day could help a manager make a judgement on their work ethic, but this data cannot be used solely to make the decision as writing more code **(5)** doesn’t necessarily mean a developer has a higher work ethic than a developer who writes less code. The manager must rely on other factors (e.g attendance at team meetings) and their own intuition to deduce a programmer’s work ethic.

Security Reasons

This is a less obvious reason to record data. **(6)** The monitoring of developers is important for a firm to prevent its intellectual property. **(6)** Recording a developer’s day to day workings acts as a security barrier and method of protection for the software developed by a firm. New software could be exposed to a competing firm which **(6)** wouldmajorly jeopardise the work that has been done by the software engineering company. While software engineering companies will have other security measures in place, monitoring their developers is essential to protecting the valuable information and code they are building for clients.

Measuring product metrics

As stated earlier on product metrics include data such as:

* How many times they commit to a repository.
* How many lines of code they write in a session.
* How many times they utilise a test suite.

This data is clear and quantitative **(2)** but can be a time expensive task. This data can also be inaccurate if it must be inputted **(1)** manually and is not inputted by automation. **(2)**The problem is that results uncovered from this data can be very inaccurate and can be an unfair representation of a developer’s work ethic. What’s more, **(2)**software projects are usually under tight time constraints and developers would rather not perform tasks such as manual inputting data to record performance as it does not deliver immediate results. Another issue is that if developers know they are being assessed on product metrics (e.g code length, commits), they will simply alter their code to make it appear that they are doing more work. Examples include putting more line breaks in their code or committing more often without actually adding anything valuable. **(1)** When this happens, the data recorded is no longer of any use as it is biased and in accurate. **(2)** Measuring the amount of bug fixes a developer makes appears to be a very good way to monitor performance. The results are clear and while some bug fixes may take longer than others to fix, analysing this data is quite straightforward and should return an unbiased result. A programmer cannot alter their code to make it appear like they are doing more work when it comes to fixing bugs whereas, when we measure how many lines of code they write, a developer can easily just put in more line breaks to give a perception that they are doing more work than others.

What is **(2)** important when we measure metrics is the quality of the content that a developer adds. Code written must add value to the project. When we measure data such as how many lines of code a developer writes, we must be aware that the quality of the work done is not being measured. A developer who writes less code or commits less often than another developer may actually be producing higher quality code. So while measuring product metrics can be useful in measuring a software developer’s performance, it is essential that the quality of their work is monitored too.

Measuring process metrics

**(2)**Process metrics include the following:

* Effort required.
* Production time
* Editing time.

Some process metrics are hard to measure such as effort required, but the times can be measured quite easily. The problem is that all projects are different **(2)** and there is no standardized time we can use as a benchmark.  **(3)** For agile and lean processes, measuring metrics such as leadtime which is the length of time a team takes to go from an idea to software developed are good indications of a team performance. This looks at the development team as a whole which has its advantages and drawbacks. A firm does not have the ability to assess and critique individual performance but as software development tasks are nearly always team based, the analysis received from data on team performance could potentially be more useful than the analysis based on individual data.

Available services and software to measure software engineering

There have been many different types of software developed to measure performance.

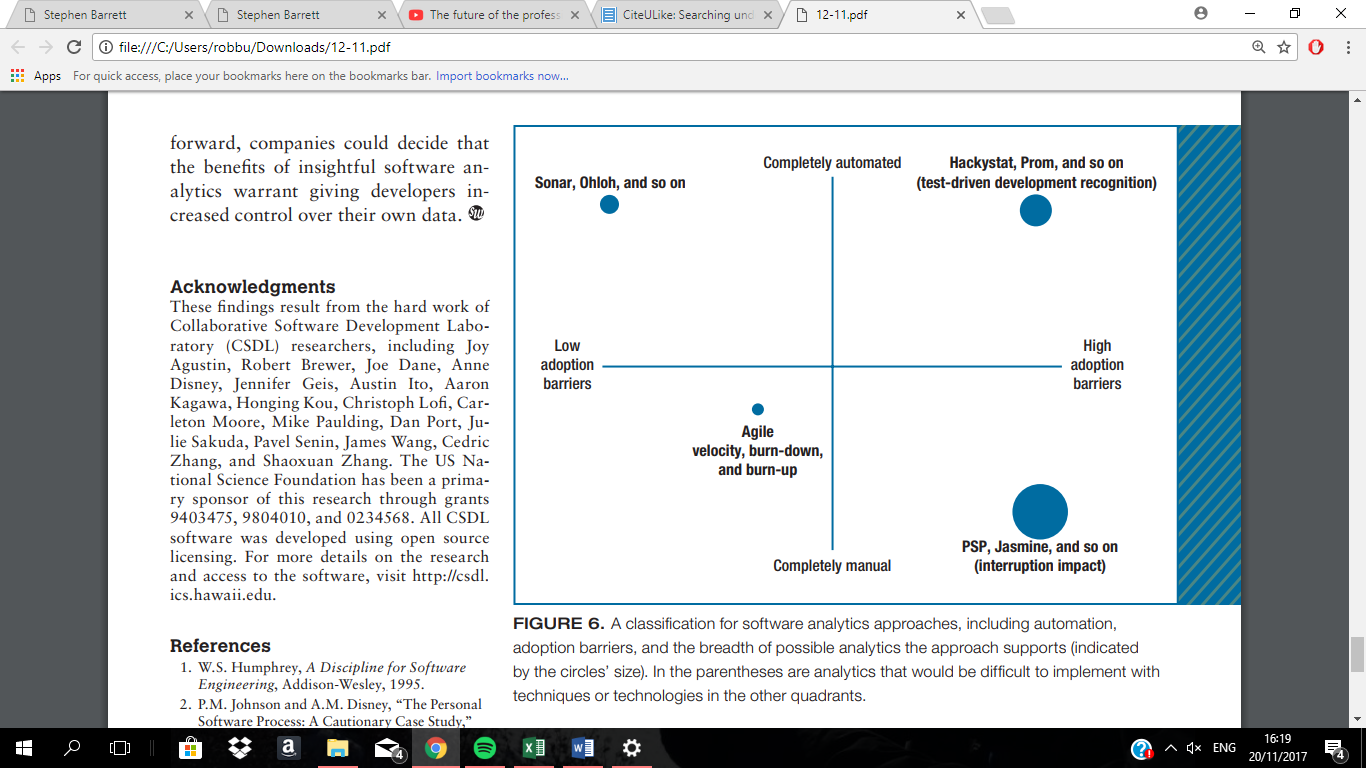
Software

When a company chooses an analytics software program to use, there are 3 trade-offs that are determined by their choice **(1)**.

(1) The degree of automation and the level of overhead developers and management incur to obtain the analytics.

(2) The barrier to adoption incurred by the technique, which could be social or political.

(3)The technique or technology’s level of generality.



<http://www.citeulike.org/group/3370/article/12458067>

The above graph illustrates the trade-offs made. **(1)**While collecting the data manually can return a lot of information, it interrupts a programmer’s ability to do work as they have to record the data themselves. What’s more, **(1)** data being recorded manually is liable to human error which can result in incorrect results. On the flip side, when completely automated software such as Hackystat are used there are different advantages and disadvantages. **(1)**Automated software is time and cost-effective and does not interrupt developers while they work however, there is a social issue as some developers are uncomfortable with their data being recorded without them knowing what exact data is being recorded at any time. To go into more detail about the advantages and disadvantages of automated and manual data analytical software programs, I will focus on two different software packages; Personal software process (PSP) and Hackystat.

Personal software process

Personal software process (PSP) is a software analytical tool which was

# - Searching under the streetlight for useful software analytics

by: [*Philip M. Johnson*](http://www.citeulike.org/group/3370/author/Johnson:PM) - <http://www.citeulike.org/group/3370/article/12458067>

(2)– **Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data -***A. Sillitti, A. Janes, G. Succi, and T. Vernazza*

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.6806&rep=rep1&type=pdf>

(3)**Tech Beacon**

<https://techbeacon.com/9-metrics-can-make-difference-todays-software-development-teams>

(4)**Java world**

https://www.javaworld.com/article/2868461/testing-debugging/how-to-and-how-not-to-measure-programmer-productivity.html

(5) **Stackify.com – measuring software development productivity**

<https://stackify.com/measuring-software-development-productivity/>

(6) **Protecting intellectual property**

https://www.csoonline.com/article/2116453/employee-protection/nine-steps-to-help-protect-software-code-intellectual-property.html