Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

# Software Engineering Measurement Report

**Koh Li Hang**
**18313798**

**11/27/2020**

# Software Measurement

## INTRODUCTION

Software Measurement run an important role in a software developing process, it helps to express the quality of a software. As we know, in software engineering, the measurement is hard to take part and not precise. However, the companies and researchers accept all kinds of measure which is better than zero measure.

Anyways, the measurement allows the acquisition of information that can be used for developing theories or work models that can improve the efficiency in software development industry.

There are some points that we can use from measuring in software, which are really useful in these modern days. [1]

- **Characterization** gather information about the characteristic of software processes and products, to have a clearly image on the current progress
- **Tracking** keep tracking on some characteristic during development, make sure those characteristics are under control
- **Evaluation** judge the characteristics of a software process or products, based on the historical data
- **Prediction** identify a cause-effect relationship among product and process characteristics
- **Improvement** use a cause-effect relationship to identify parts of process of products that can be improve in certain parts

In my report, I will discuss the ways in which the software engineering process can be measured and assessed in terms of measurable data, the platform can be used to gather and process data to achieve the goal mentioned above. Also, the algorithms being used in this process. Finally, have a discussion on ethical concern on this.

# Classification of Software Measurement

## TYPES

## Direct Measurement

This type of measurement can be measured without the involvement of any other entity or attribute.

Here are the common direct measures used in software engineering.

- Length of source code (SLOC or LOC)
- Duration of testing purpose by elapsed time
- Number of defects discovered during the testing process by counting defects (Defect measures)
- The time of a programmer spends on a program (Programmer time cost)

### Source Line Of Code (SLOC)[2]

Source lines of code is a software metric used to measure the size of a program by counting the number of lines in the program's source code. Usually, it is used to predict the effort that will be required to develop a project, as well as to estimate the programming productivity or effort once the project is completed.

### SLOC Measurement Methods

There are several types of SLOC measures, but we are going to focus on 2 major types, which are physical SLOC (LOC) and logical SLOC (LLOC).

- **Physical SLOC (LOC):** A count of lines in the text of the program's source code including comments line (Blank lines also included, except the lines of code consists more than 25% of blank lines)
- **Logical SLOC (LLOC):** usually use to measure the number of "statements", but its specific definition is tied to specific computer languages (for examples, one of the LLOC for C-like programming languages is the number of statement-terminating semicolons)

**Here's an example to explain the difference between them:**

```
for (i = 0; i < 100; i += 1) printf("hello");/* How many lines of code? */
```

Components:

- 1 LOC (physical line of code)
- 2 LLOC (logical line of code) [for statement and printf statement]
- 1 Comment line

**Here's another example to explain the difference between them:**

```
for (i = 0; i < 100; i += 1)
{
    printf("hello");
} /* Now how many lines of code is this? */
```

Components:

- 4 LOC (physical line of code)
- 2 LLOC (logical line of code) [for statement and printf statement]
- 1 Comment line

## Duration of testing purpose by elapsed time

From here, we are able to check the efficiency of the codes and also the tests. Normally, this can be done in IDE.

## Defect measures [3]

There are 4 common defects metrices are included in this type of measurement, they are defect density, defect discovery rate, cumulative discovery rate, defect counts by type.

**Defect density** in a program is typically used to reflect structural quality.

**Defect discovery** rate is a count of the number of defects being discovered over time. It is tending to follow common patterns, and it is used to assess all aspects of software quality depending on the type of testing.

**Cumulative discovered defects** are the number of defects discovered for a specific piece of project summed up and presented over a period of time. The time increments used are typically a reflection of the type of work being done.

**Defect counts by type** can be used to monitor process, structure or functional aspects of quality depending on when data on defects is collected and the taxonomy used to categorize defects. This simple measure always the starting point for other defect measures, and its simplicity makes it easy to start and stop.

## Programmer time cost

This is pretty easy to understand, which it reflects the programmer abilities, project size.

# Classification of Software Measurement

## TYPES

## Indirect Measurement

This type of measurement can be measured in terms of any other entity or attribute.

Here are the common indirect measures used in software engineering.

$$Programmer\ Productivity = \frac{LOC\ produced}{Person\ months\ of\ effort}$$

$$Module\ Defect\ Density = \frac{Number\ of\ defects}{Module\ size}$$

$$Defect\ Detection\ Efficiency = \frac{Number\ of\ defects\ detected}{Total\ number\ of\ defects}$$

$$Requirement\ Stability = \frac{Number\ of\ initial\ requirements}{Total\ number\ of\ requirements}$$

$$Test\ Effectiveness\ Ratio = \frac{Number\ of\ items\ covered}{Total\ number\ of\ items}$$

$$System\ spoilage = \frac{Effort\ spent\ for\ fixing\ faults}{Total\ project\ effort}$$ [4]

Indirect measure is the product includes functionality, quality, complexity, efficiency, reliability, maintainability, and many other quality attributes.

It is used when the characteristic to be assessed cannot be measured directly. For instances, 'quality' is had to be measured directly, but can be measured with other software characteristics.

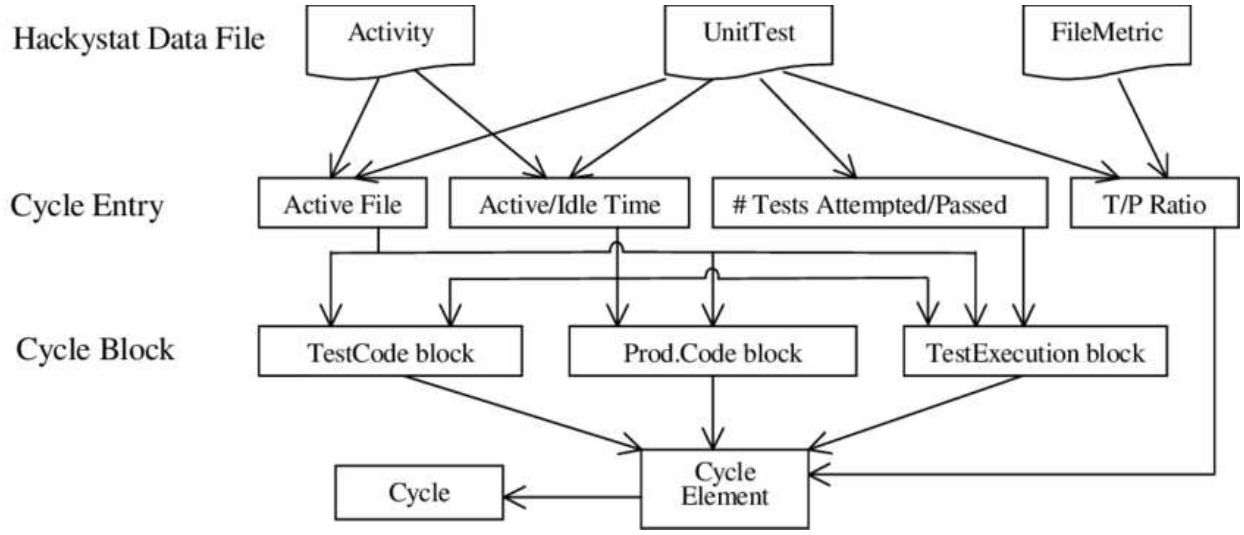
Software is not a tangible entity, we can still apply direct measures on it, but commonly indirect measurements are preferred.

# Platforms for SE Measurements

## PLATFORMS

### Hackystat [5]

**Here is a brief progress diagram on how Hackystat works:**




[6]

Hackystat is an open source framework for collection, analysis, visualization, interpretation, annotation, and dissemination of software development process and product data.

Hackystat users usually attach software "sensors" to their development tools. This will unobtrusively collect and send raw data about development to a web service call the Hackystat Sensor Base for storage. The data can be queried by other web services to form higher level abstractions of this raw data, to generate visualizations of the data, abstractions or annotations.

Hackystat services are designed to co-exist and complement other components in the Cloud internet information system and services available for modern software development.

# GitPrime [7]

**GitPrime demonstration diagram:**

[8]

GitPrime is a reliable productivity analytics software that analyzes data from codebases to deliver in-depth reports on the progress of the software development process. This platform is renowned for its power to display a comprehensive overview of the entire software development process.

It helps managers to visualize the activities in real-time, plus the history data is availed for reference purposes. GitPrime bridges communications gaps and allows managers to lead teams with facts for increased productivity.

Here is an overview of GitPrime features:

**Historical data, Data visualization, Work pattern highlighting, Email digest, Productivity reporting, Productivity graphs, Keyword search, Codebase analysis**

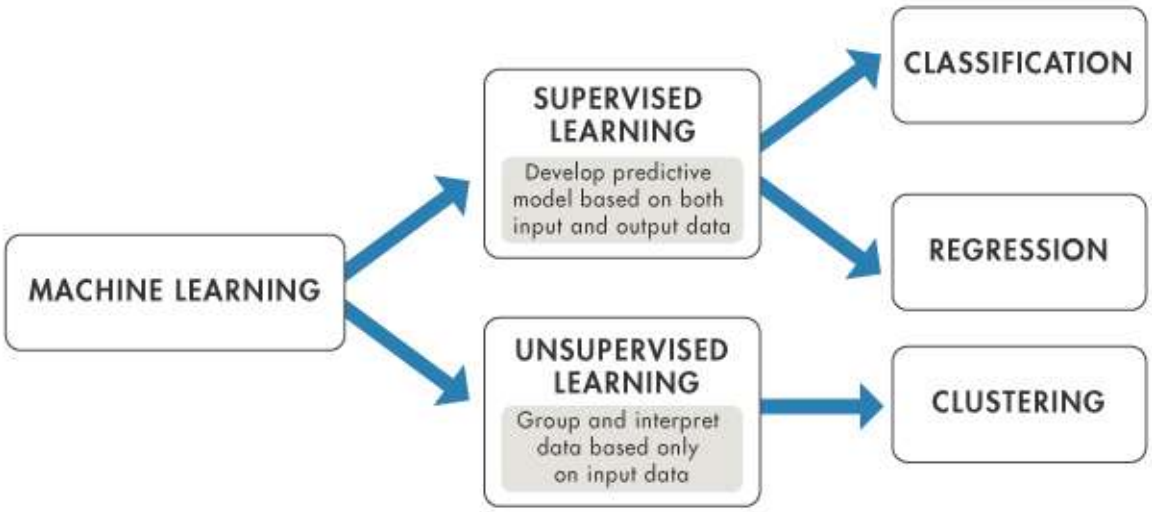
# Algorithms for SE Measurements

## ALGORITHMS

## Machine Learning

Undeniable that the software development industry is adopting machine learning for transitioning modern day software systems towards highly intelligent and self-learning systems.

Here we are going to talk about 3 major types of ML, and briefly tell how they are used in software engineering measurements. They are **Supervised Learning**, **Unsupervised Learning**, and **Reinforcement Learning**.

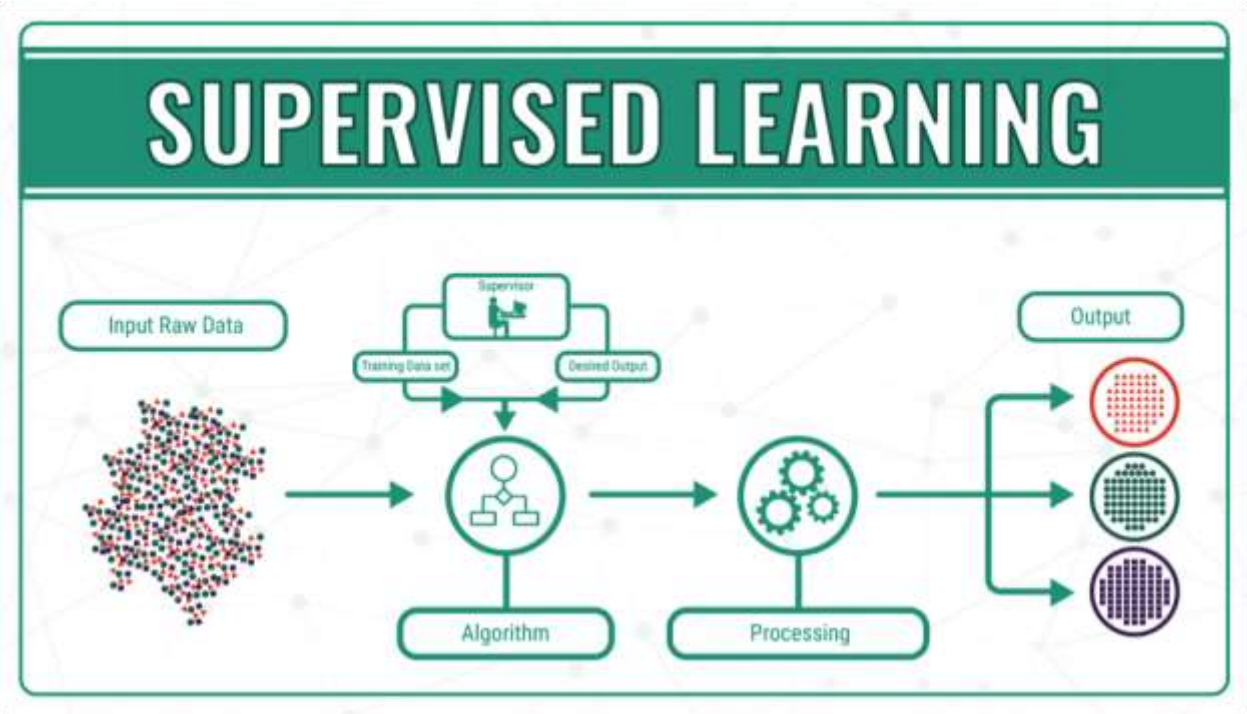



[9]

## Supervised Learning

Supervised learning trains the machine by using the data which is well-labeled. This means that the data being inputs is already tagged with the correct output. It learns in a way by taking place in the presence of supervisor, then evolve.[10]

**Here's a diagram which shows how the supervised learning model works:**




[11]

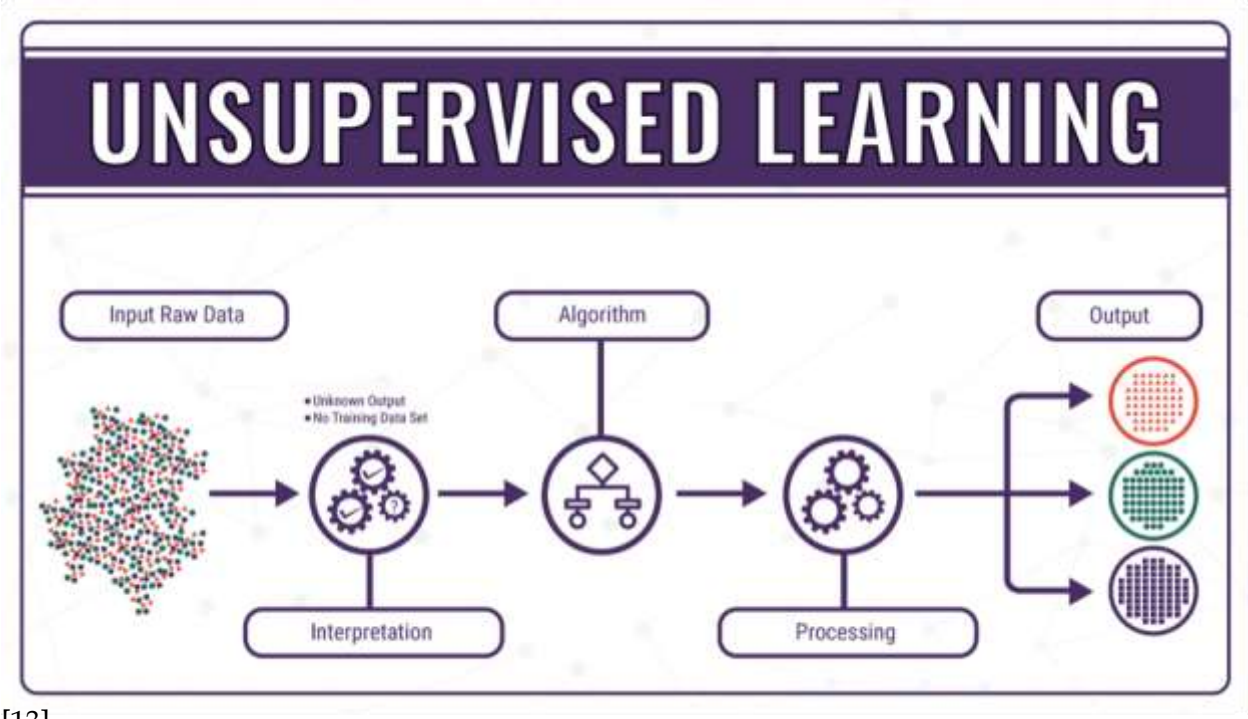
Supervised learning is also an approach to create artificial intelligence (AI). The AI system is specifically told what to look for, thus the model is trained until it is able to detect the underlying patterns and relationships, enable it to produce outputs when presented with data which is never-before-seen.

This algorithms type can be used in developing an artificial intelligence system for Design Pattern Prediction, Development Effort Estimation, Testing Effort Estimation and furthermore.

# Unsupervised Learning

Unsupervised Learning is a type of machine learning that the users will not need to supervise the model. This means that it allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with the data which is unlabeled.[12]

**Here's a diagram which shows how the unsupervised learning model works:**




[13]

Unsupervised learning algorithms allow users to perform more complex processing tasks compared to supervised learning. Although, unsupervised learning can be more unpredictable compared with other natural learning methods. Unsupervised learning algorithms include clustering, anomaly detection, neural networks, etc.

There are some applications in SE measurement that used unsupervised learning. For instances, anomaly detection can discover unusual data points in your dataset. It is useful for finding fraudulent transactions or defects. Besides that, Latent variable models are widely used for data pre-processing. Like reducing the number of features in a dataset or decomposing the dataset into multiple components.

# Ethical Concerns

## CONCERNS

## Privacy Concern

### Data Gathering

As the most important part in software engineering measurement is data gathering, how to deal with the privacy concern is always the first priority. The data will include a person daily work life, work performance, and more personally information. If the data is leaked might cause huge problems, or indirectly causing social problems. Therefore, a fully structured law and protection must be used when we are working on some sensitive personal details.

### Data Analysis

Besides that, as mentioned above we are now using Machine Learning Models in Software Engineering Measurements. Somehow, the models or the algorithms outputs is not 100% correct. For example, the stock market forecasting machines can output nearly 99% of accuracy of the successful prediction, but there's still 1% that can make a huge difference. Furthermore, what is there's a work efficiency rate prediction on software engineer? Actually, most of the big companies are using this kind of system to keep their company operations. Although this might help the company to achieve a huge success, but seems for now, instead of working for a company or a boss, performing well under an algorithm is more important. So, letting the employees understand how the algorithm works and what actually the data being used is very important.

# Citations

## LINKS

[1] https://stackify.com/track-software-metrics/
[2] http://www.projectcodemeter.com/cost_estimation/help/GL_sloc.htm
[3] https://tcagley.wordpress.com/2016/07/14/five-types-of-defects-measures-useful-during-development/
[4] https://www.tutorialspoint.com/software_quality_management/software_quality_management_measurement_models.htm
[5] https://hackystat.github.io/
[6] https://www.researchgate.net/figure/Recognizing-programming-cycles-using-Hackystat-data-types_fig1_221592672
[7] https://reviews.financesonline.com/p/gitprime/
[8] https://www.transpire.com/insights/blog/data-driven-leadership-goodbye-pointless-kpis-hello-meaningful-metrics/
[9] https://medium.com/@chisoftware/supervised-vs-unsupervised-machine-learning-7f26118d5ee6
[10] https://www.guru99.com/supervised-machine-learning.html
[11] https://www.educative.io/edpresso/supervised-learning-algorithms
[12]
[13] https://uk.mathworks.com/help/stats/machine-learning-in-matlab.html
[14] https://www.guru99.com/unsupervised-machine-learning.html