# MEASURING SOFTWARE ENGINEERING

Sai Bala Subrahmanya Lakshmi Kanth Rayanapati

kanthras@tcd.ie

19304511

## Introduction:

Software Engineering was a relatively new field and the need for it was not recognized until the mid 1960s. By the late 1960s it became apparent that the software development processes followed were inefficient and new and more systematic approaches were required. The phrase software engineering, which refers to highly disciplined and systematic approach to software development and maintenance, was first invented by the mathematician Margaret Hamilton and was more openly discussed and given an initial boost at the NATO sponsored meetings in 1968 and 1969.

Bad software engineering processes can cause catastrophic failures. One such incident was the loss of NASA's Mars climate orbiter in 1998, which was believed to have crashed into the Mars' atmosphere because of an error in the conversion of units. In some cases, bad software engineering practices in the field of medicine can cause harm to a persons' life. One such example was the Theract-25, which was a radiation therapy machine that was involved in 6 accidents where patients were given excessive overdoses of radiation. Incidents such as this were the result of bad software engineering practices. So, it became consequently clear that there is a need to measure software engineering.

Measuring software engineering helps in generating a higher quality code by detecting potential software flaws in early stages of development by unit testing. It helps in decreasing the cost and the product development time. It

can also be used to identify high performing and the low performing employees and reward them accordingly.
In this report, we will be viewing the various of measuring software engineering, then we will focus on the platforms that can be used to measure the engineering activity and the various computations that can be done over the data and finally we will focus on the ethical and legal issues surrounding this process.

## Software Engineering Measures:

There are various software metrics that can be used to measure software engineering activity and large amount of information can be acquired on a single software developer.

## Source Lines of Code (SLOC):

Source Lines of Code also called Lines of Code is the simplest and easiest metric used to measure the productivity of a software engineer. Lines of code is a software metric which measures the productivity of a software engineer by counting the number of lines in the program’s code base.
There are two kinds of measures while using Lines of code metric: physical Lines of code and Logical Lines of code. Physical LOC counts the number of lines in the software engineer’s source code excluding the comments. On the other hand, Logical LOC is a bit more complicated as it counts the number of executable statements in the software engineer’s code. Different programming languages have different definitions for Logical LOC as executable statements are written different syntax. For example, in the languages like java and c, Logical LOC is measured by measuring the number of semi-colons.
Physical and Logical LOC metrics can be different from each other as Physical LOC metrics change relative to logically irrelevant formatting and

style whereas Logical LOC metrics do not. We can observe the following the example below.

Example:

```
for(int i=0; i<10 ; i++)
{
    System.out.println("Hello world!");
}//Comment
```

In the above piece of code we have,

- Four Physical Lines of code
- Two Logical Lines of code
- One comment line

The code below has the same functionality as the piece of code above but has different format to it.

```
for(int i=0; i<10 ; i++) System.out.println("Hello world!"); //Comment
```

In this code we have,

- One Physical Lines of code
- Two Logical Lines of Code
- One Comment line

## <u>Number of Commits:</u>

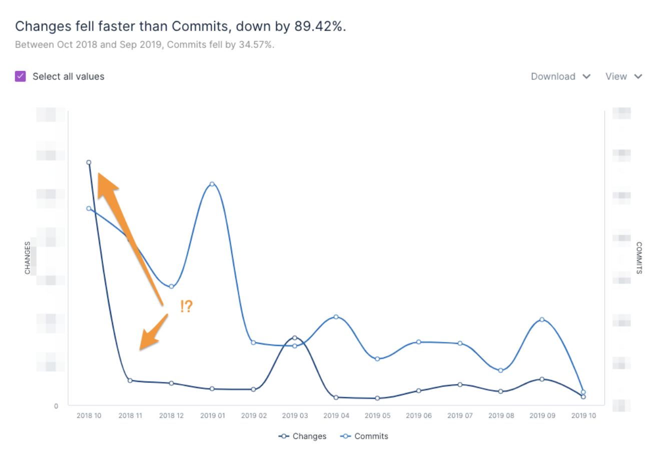


*Figure 1: The graph above shows the difference between the number of changes made and commits (Taken from Ref [1])*

Another way to measure the productivity of a software engineer is to measure the number of commits they have done to their code. Similar to Lines of code, even this metric is not very accurate method to measure the productivity of a Software engineer. A team with high number of commits could deliver the same outcome a team with a smaller number of commits. The main issue with counting the commits is that it does not indicate the progress done. This method encourages small and unnecessary commits.

## Cycle Time:

Cycle time is one of the most important metric measures to measure the productivity of a software engineer. The entire time that elapses between the start and end of a work (a bug fix, code, …) is called Cycle time. It can also be defined as the period during which a team or a software engineer is working on developing a part of the software. Cycle time measures the accuracy of a particular team or a software engineer at delivering a software. This will help in increasing the efficiency/ productivity of a software engineer.

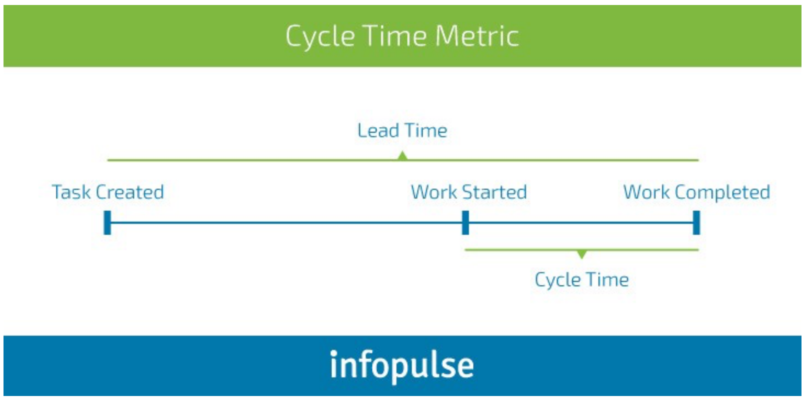


*Figure 2:The image above (Taken from Ref [2]) shows a graphical representation of cycle time.*

## Computational Platforms Available

There are many computational software which visualize the data and make it easy to analyze the data and measure the productivity of a software engineer.

### Pluralsight Flow / GitPrime

Pluralsight Flow previously known as GitPrime is a metric visualization software which uses data from any git-based code repository (e.g.: GitHub, GitLab, BitBucket, SVN) and analyses the data collected. Pluralsight Flow provides detailed information and graphs on the metrics like pull requests, Lines of code, changes made, commits and many more. This data can then be used to show the productivity of a software engineer. Pluralsight Flow also gives detailed information on type of the work, Churn, Efficiency and the impact.

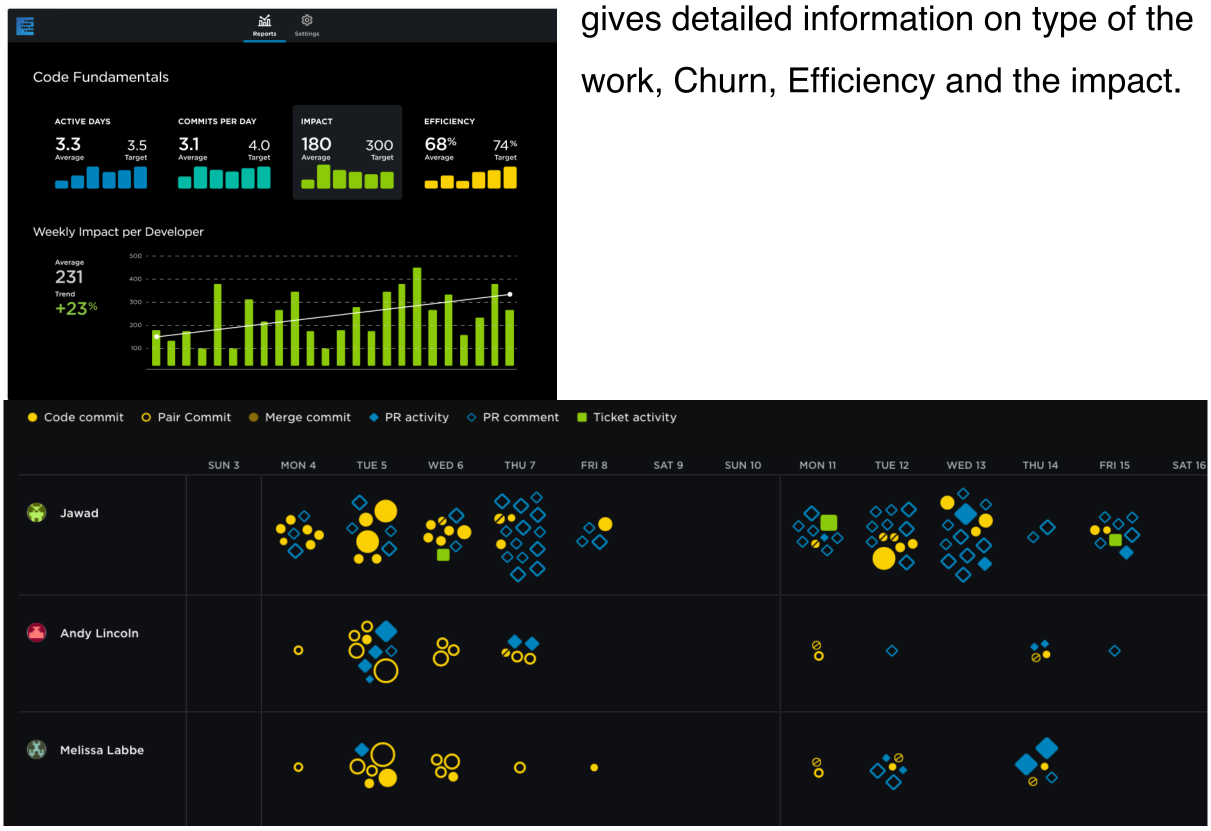
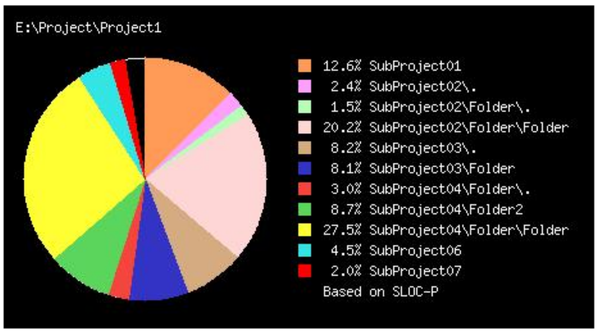
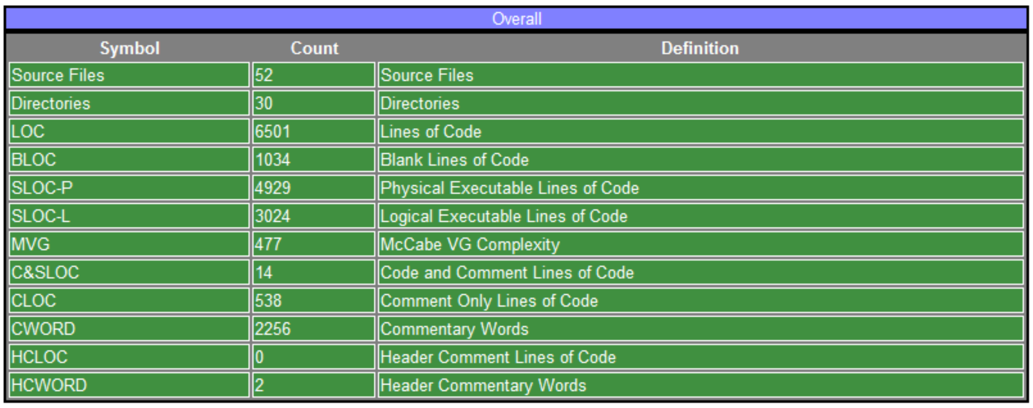

Reports
Settings
Code Fundamentals
ACTIVE DAYS 3.3 Average 3.5 Target
COMMITS PER DAY 3.1 Average 4.0 Target
IMPACT 180 Average 300 Target
EFFICIENCY 68% Average 74% Target
Weekly Impact per Developer
Average 231
Trend +23%
Code commit
Pair Commit
Merge commit
PR activity
PR comment
Ticket activity
SUN 3 MON 4 TUE 5 WED 6 THU 7 FRI 8 SAT 9 SUN 10 MON 11 TUE 12 WED 13 THU 14 FRI 15 SAT 16
Jawad
Andy Lincoln
Melissa Labbe

*Figure 3: Output of the software taken from Ref [3]*

## **SLOCMetrics**

SLOCMetrics also known as LOCMetrics is one of the most popular computational tools to measure the Source Lines of Code to measure the productivity of a software engineer. SLOCMetrics supports the programming languages like c#, c++, java, SQL, python and JavaScript. It is a very simple tool to use. It takes in the file to perform the analysis on and gives a very detailed analysis of the Lines of Code metric.



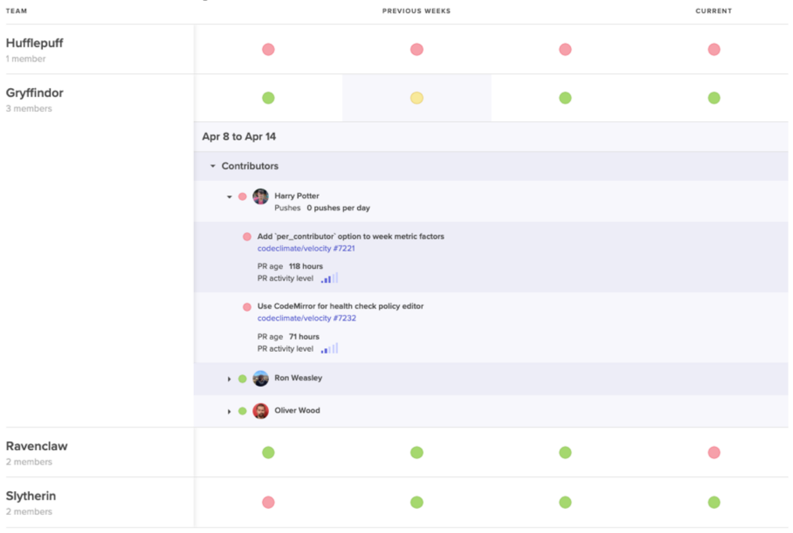


| Overall | | |
|---|---|---|
| **Symbol** | **Count** | **Definition** |
| Source Files | 52 | Source Files |
| Directories | 30 | Directories |
| LOC | 6501 | Lines of Code |
| BLOC | 1034 | Blank Lines of Code |
| SLOC-P | 4929 | Physical Executable Lines of Code |
| SLOC-L | 3024 | Logical Executable Lines of Code |
| MVG | 477 | McCabe VG Complexity |
| C&SLOC | 14 | Code and Comment Lines of Code |
| CLOC | 538 | Comment Only Lines of Code |
| CWORD | 2256 | Commentary Words |
| HCLOC | 0 | Header Comment Lines of Code |
| HCWORD | 2 | Header Commentary Words |

*Figure 4: The output of SLOCMetrics on the analysis of source files. Image taken from Ref [4]*

## Velocity 2.0

Velocity 2.0 is an upgrade of the software Velocity by code climate. Velocity 2.0 is a very versatile software and allows teams to customize Metrics, Algorithms, Thresholds, Reports, Dashboards and Permissions. This software provides insights on the data which are similar to Pluralsight Flow. Velocity 2.0 can provide a clear insight on the data analyzed which will help in measuring the productivity of a team or a software engineer.



| TEAM | PREVIOUS WEEKS | | | CURRENT |
|---|---|---|---|---|
| Hufflepuff<br>1 member | 🔴 | 🔴 | 🔴 | 🔴 |
| Gryffindor<br>3 members | 🟢 | 🟡 | 🟢 | 🟢 |
| Ravenclaw<br>2 members | 🟢 | 🟢 | 🟢 | 🔴 |
| Slytherin<br>2 members | 🔴 | 🟢 | 🟢 | 🟢 |

Apr 8 to Apr 14

Contributors

Harry Potter
Pushes 0 pushes per day

Add `per_contributor` option to week metric factors
codeclimate/velocity #7221
PR age 118 hours
PR activity level

Use CodeMirror for health check policy editor
codeclimate/velocity #7232
PR age 71 hours
PR activity level

Ron Weasley

Oliver Wood

*Figure 5: This graph is the output of the software taken from Ref [5]*

## Algorithm Approaches:

### COCOMO Model:

Constructive Cost Model also known as COCOMO Model is a regression model based on Source Lines of Code metric measure. It was proposed in 1981 developed by the American Software engineer Barry W. Boehm. COCOMO model is one of the most widely used model for software estimation. The COCOMO model is used to estimate the cost, effort and the amount of time required to complete a software project based on the size of the software.

For the prediction of cost estimation, the software projects that employ COCOMO models have been proposed depending on the level of precision and correctness required. Based on the factors like number of lines of code, size of the team, developer experience, innovation and deadlines COCOMO model can be categorized into 3 categories: Organic, semi-detached and embedded.

Further COCOMO models are divided into 3 types to aid the software manager to choose how precisely cost estimation is to be performed. The different types are:

- Basic COCOMO Model
- Intermediate COCOMO Model
- Detailed COCOMO Model

Basic COCOMO model is the base level which is used to calculate the cost of the software based on rough estimations. Intermediate COCOMO Model is the next level and is built on the Basic COCOMO model. It considers the cost drivers which will help in much better estimations. The third layer, The Detailed COCOMO model is built on both the Basic and Intermediate COCOMO models. The Detailed COCOMO model produces the most accurate estimations.

## <u>Machine Learning Analysis:</u>

"**Machine learning** (**ML**) is the study of computer algorithms that can improve automatically through experience and by the use of data." (From Ref [9]) In other words, Machine learning is a subgroup of Artificial Intelligence which uses data and algorithms to train the Machine Learning model. Machine learning is one of the most popular and widely used algorithms to tackle problems. Machine Learning has numerous applications including facial recognition software, self-driving cars and many more. Machine Learning methods can also be used to measure the productivity of a software by examining the code and improving its performance.

There are three types of Machine learning algorithms to train the data

- Supervised Learning Methods
- Unsupervised Learning Methods
- Reinforcement Learning

## <u>Supervised Learning Model:</u>

Supervised Learning Methods assign labels to observations using a structure provided within the data. In other words, in Supervised learning we divide the original data set into training data and testing data. We then use the labelled training data to train our algorithm. This algorithm can then be used to analyze the testing data set when required.

Supervised learning models are used in a variety of recognition software. One example is the software which recognizes the type of the vehicle on the road. First, we can train the

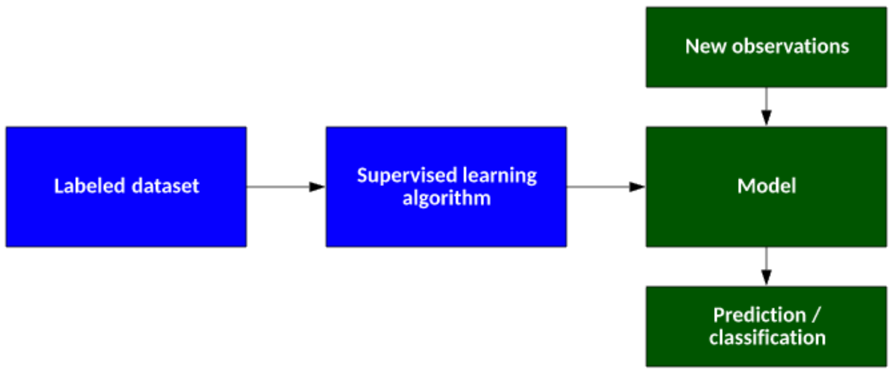


*Figure 6: Graphical representation of the model taken from Ref [6]*

model by passing images of cars, trucks, motor bikes and labelling them accordingly. Then this trained model will group a new vehicle accordingly to the respective group based on the similarities.

## Unsupervised Learning Model:

Unsupervised learning methods rely solely on the internal structure of the data to explain the data in a reduced number of dimensions or to assign labels to the observations. In other words, this algorithm when provided with a large set of unlabelled data, it will group similar observations into similar groups.

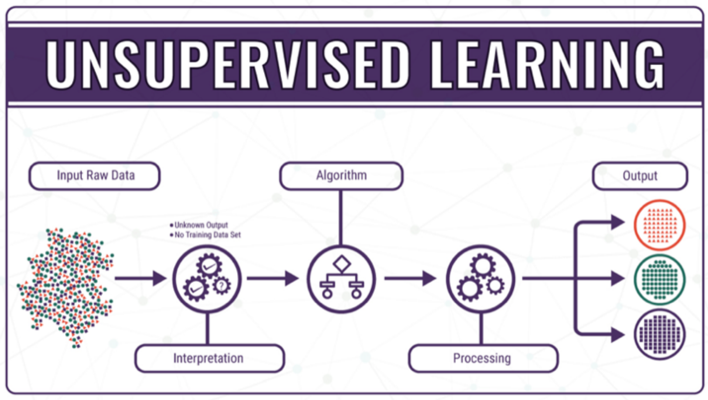


Figure 7: Graphical representation of the method taken from Ref [7]

Unsupervised Learning models are used to group people with similar characteristics and personalities to generate customer profiles for more effective marketing.

## Reinforcement Learning:

Reinforcement Learning is a machine learning algorithm which interacts with its environment and continuously learns based on the feedback it keeps getting. The feedback given by the environment is either positive or negative. Positive feedback is given when the algorithm performs a desirable action and negative feedback for undesirable actions. Reinforcement learning is one of the hardest models to implement. One of the most famous applications

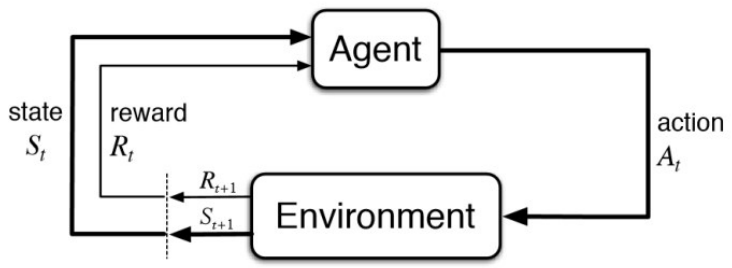


Figure 8: Graphical Representation of the method taken from Ref [8]

of the Reinforcement Learning model is the Pac-man game. This model is also used in self driving cars.

## Ethical Concerns

There are a variety of ethical concerns regarding measuring the productivity of a software engineer. The first area of concern is the collection of data of the software engineer for measuring productivity. This area wouldn't be an ethical concern in most of the cases as this data is provided by the software engineer himself. But in some cases, the software engineer can unnecessarily increase the number of lines of code or the number of commits to look more productive than others when either Source Lines of Code (SLOC) or number of commits are used as a metric of measuring the productivity of a software engineer. It could be an area of concern if the data collected is personal information of the employer and gives detailed overview of the employee's personal life. Even though this data can be used by some metrics to analyze the productivity of an engineer I believe an employee must have privacy regarding his personal life. Also, it could pose a very high risk if the information stored is leaked or accessed by someone as this is the sensitive information of the employer. Data breaches have become common in the recent years. One such example is the data breach of 700 million LinkedIn users in June of 2021.
The other area of concern is the use of improper metrics and being over dependent on the output of the metrics to measure the productivity of a software engineer. As we saw above, being over dependent on metrics such as Source Lines of Code, number of commits and many more can influence the software engineer to use bad practices to increase their productivity.

## References

https://betterprogramming.pub/looking-for-metrics-in-all-the-wrong-places-db63ca81d4e9 [1]

https://medium.com/@infopulseglobal_9037/top-10-software-development-metrics-to-measure-productivity-bcc9051c4615 [2]

https://www.pluralsight.com/product/flow/code [3]

https://www.pluralsight.com/blog/teams/lead-your-team-to-better-mob-programming [3]

https://www.cheonghyun.com/blog/120 [4]

https://codeclimate.com/blog/velocity-2-is-here/ [5]

https://developer.ibm.com/articles/cc-supervised-learning-models/ [6]

https://technative.io/why-unsupervised-machine-learning-is-the-future-of-cybersecurity/ [7]

https://www.analyticsvidhya.com/blog/2021/02/introduction-to-reinforcement-learning-for-beginners/ [8]

https://en.wikipedia.org/wiki/Machine_learning [9]

https://mediaweb.saintleo.edu/Courses/COM430/M1Readings/A%20Brief%20History%20of%20Software%20Engineering.pdf

https://en.wikipedia.org/wiki/History_of_software_engineering#1945_to_1965:_The_origins

https://www.nasa.gov/50th/50th_magazine/scientists.html

https://www.buildingbettersoftware.com/blog/the-impact-of-poor-software-quality/

https://en.wikipedia.org/wiki/Therac-25

https://www.cs.purdue.edu/homes/cs307/slides/lecture14_2up.pdf

https://www.getclockwise.com/blog/measure-productivity-development

http://www.projectcodemeter.com/cost_estimation/help/GL_sloc.htm

https://en.wikipedia.org/wiki/Source_lines_of_code#Usage_of_SLOC_measures https://www.usehaystack.io/blog/software-development-metrics-top-5-commonly-misused-metrics

https://codeclimate.com/blog/software-engineering-cycle-time/

https://www.javatpoint.com/cocomo-model

https://www.geeksforgeeks.org/software-engineering-cocomo-model/?ref=lbp

https://medium.com/@warakornjetlohasiri/cocomo-a-regression-model-in-procedural-cost-estimate-model-for-software-projects-65ab5222a1f5

https://en.wikipedia.org/wiki/COCOMO

https://www.altexsoft.com/blog/unsupervised-machine-learning/

https://scand.com/company/blog/how-machine-learning-is-changing-software-development/

https://www.ibm.com/cloud/learn/machine-learning

https://pioneerlabs.io/insights/the-three-types-of-machine-learning-algorithms/

https://en.wikipedia.org/wiki/Reinforcement_learning

https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning

https://stackshare.io/stackups/github-vs-gitprime

https://www.pluralsight.com/product/flow

https://help.pluralsight.com/help/metrics

https://help.pluralsight.com/help/chapter-2-flow-import-and-manage-your-data