Measuring Software Engineering Report

Name: - Kinjal Bhattacharyya

Roll no.: - 21344426

Contents

Introduction

‘Software metrics’ is a collective term used for describing the variety of activities related to measurement in software engineering. In this report, I have described some of the metrics used for measuring productivity in software engineering and the advantages of some metrics over others. I have also described the several platforms that are available today to process the large amounts of data generated through these processes in terms of performance as well as cost effectiveness. Next, I have described some of the algorithms that could be deployed for doing the required computation and the analysis. Finally, I have written about the Moral, Ethical and legal obligations of judging a developer's productivity by these metrics.

Measuring engineering Activity

Measuring Engineering activity has been in practice since the 1960s to provide the managers with information regarding the performance and efficiency of a team in software development and testing. Historically, the method often used to quantify the effectiveness of a software developer is either by Lines of code measure (LoC) or KLOC (when thousands of lines of code were involved).

However, the problem with LoC is that it wasn’t aimed at a particular measuring metric in mind as it attempted to measure both the software developer’s performance and their program quality. In other words, it was an all-encompassing measure of the different aspects of program size that focussed not only on maximizing functionality but also reducing complexity. However, the LoC measure was originally designed for assembly language code and suffered incompatibility in terms of both functionality and complexity with the variety of high-level languages that followed.

Hall and Fenton in 1972 said,

“A metric program established with a clear goal in mind is doomed to fail”

Thus, a more goal-oriented approach became necessary.

Our primary goal with metrics is of course to maximize the productivity of the engineers. However, the various metrics that can be deployed to analyse the information an organization collects can often be misleading and can negatively impact the members of that organization. Among these various metrics, the most misleading ones are often those that are focussed on the result instead of the process as outputs like lines of code, number of commits or pull request count cannot predict the productivity of the developer, especially in the quality of the code written. Thus, it is more useful to set our focus on the process and measure the developer's performance against pre-determined targets. This leads to more collaborations among team members and yields greater productivity within the organization. Metrics that are more focussed on process than output are code review turnaround time, pull request size etc.

Code Review Turnaround Time and pull request size are good metrics to measure the productivity of a team and is somewhat interconnected. With Code Review Turnaround Time, we measure the time a team takes to review a pull request. With faster response time to pull requests, teams working on a project can make faster commits on it in a more collaborative atmosphere.

However, the team making the pull request must also ensure that the pull request size is small enough so that the reviewer's ability to detect defect code is maximized. The following graph shows how the increase in the lines of code given for review can negatively impact the ability of a reviewer to identify defective code.

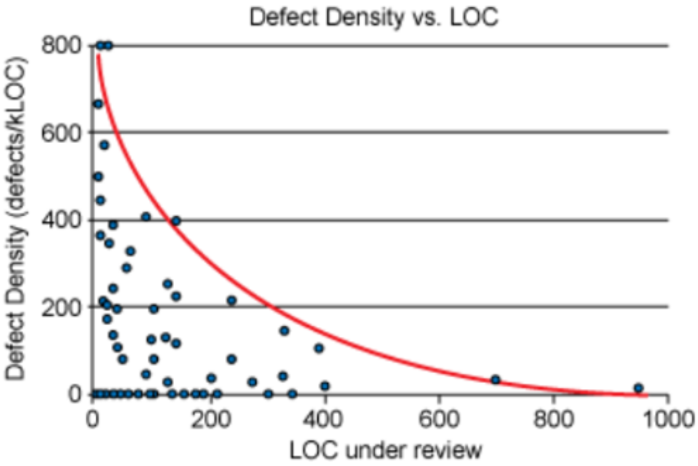

Fig 1: A SmartBear study of a Cisco Systems programming team showing their Defect density v/s LOC (Source: - https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/)

Thus, it is also important that we determine the optimal pull request size that increases performance and reduces cost in the organization.

To find the optimum pull request size that a team can work with we need to optimize transaction cost and holding cost with respect to pull request size. We can therefore treat this as an optimization problem dealing with transaction cost that includes creating, reviewing, managing, and merging a pull request and holding cost that occurs when a team keeps the pull request as a work in progress. With increase in pull request size, the transaction cost decreases but the holding cost increases. The point at which the total cost (transaction cost+ holding cost) is minimized is the optimum pull request size for that team.

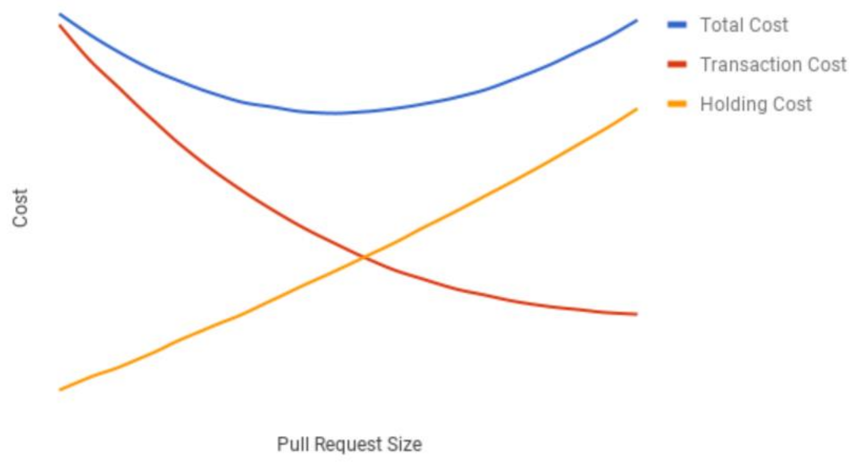

Fig 2: - Optimal Pull Request Size (Source: - https://smallbusinessprogramming.com/optimal-pull-request-size/)

Platforms for big data analytics

There are several big data platforms available with us today and choosing the right one for data processing is crucial to adapt to the growing data processing demands as well as be cost-effective. The ability of a data processing platform to adapt to the increasing size of our data determines how scalable our platform is. There are two scaling types that can be used to handle big data. They are: -

i. Horizontal Scaling – The data processing load is scaled out across multiple servers to improve processing capability.

ii. Vertical Scaling – More processors, more memory, and faster hardware are added to a single server to improve processing capability.

While there is no limit to how much a system implementing horizontal scaling can be scaled up, there is certainly a limit with vertical scaling where it can reach a point when scaling up is no longer feasible. Moreover, financial investment to do horizontal scaling is relatively less compared to vertical scaling.

Two of the commonly used platforms in Horizontal scaling are peer-to-peer networks and Apache Hadoop. While peer-to-peer networks implements grid computing, Apache Hadoop can be implemented on cloud services and can therefore utilize utility computing.

Apache Hadoop

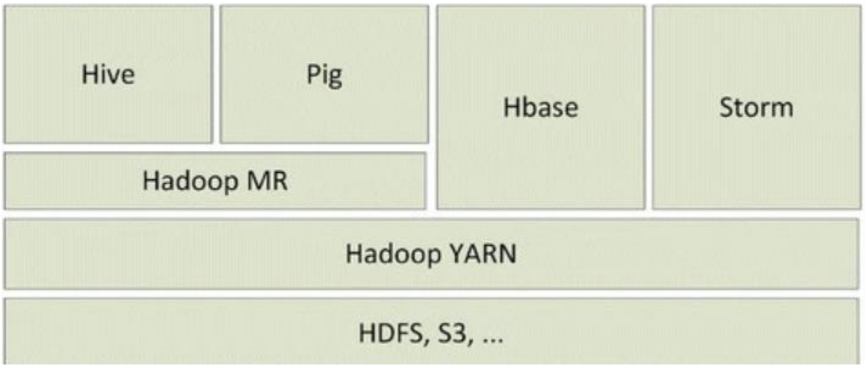

Fig 3: Hadoop stack showing different components

Apache Hadoop is an open-source framework that can be used for storing and processing very large datasets. This framework is highly fault-tolerant and can be scaled up to thousands of nodes.

Among the various components of Hadoop shown in Fig 3, HDFS is a distributed file system that provides the framework's high availability and fault tolerance while storing data across clusters of commodity machines and Hadoop YARN is the job scheduler across the clusters.

An important feature of Apache Hadoop is that it can be used atop of cloud-based services such as Amazon Web Services. This allows users to pay for the clusters only when they need them. Hadoop jobs vary in time and frequency. Therefore, while certain jobs are run frequently, some tasks like genetic processing are only run a few times in a year. The advantage of using Hadoop on a cloud platform is that one can process the workload clusters, save the results, shut down the Hadoop resources when they are no longer required, and avoid incurring unnecessary infrastructure costs.

Moreover, support of dynamic scaling of nodes on this platform is cost-effective and allows the user to create the cluster of the required size instantaneously instead of assigning resources that sit idle when there is no job for it and drives up the infrastructure cost.

Apache Spark

Apache Spark is the next generation of data processing platform developed by researchers at UC Berkley. Apache Spark is an alternative to Apache Hadoop that offers better performance and overcomes the latter's disk I/O limitations.

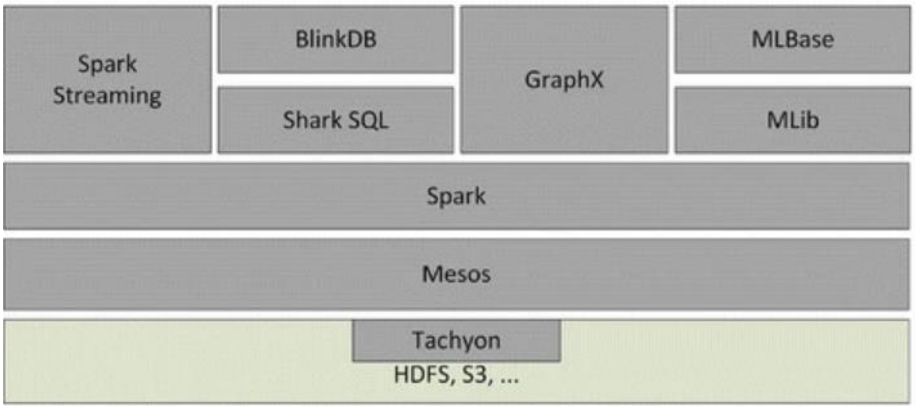

Fig 4: Different components of BDAS

Spark runs on BDAS or Berkley Data Analytic Stack which is a proposed data processing stack through which spark can provide its improved features. At the lowest level of this stack is HDFS based Tachyon file sharing system. Tachyon is a distributed file sharing system that is Fault tolerant and provides file sharing at memory speed. It detects the files that are most often used in the file system and caches them in the memory to minimize disk access. This gives it a significant advantage over Hadoop HDFS in terms of performance by utilizing memory more aggressively.

The second component of BDAS is Mesos that offers resource isolation and file sharing across distributed frameworks. It supports spark on a dynamically shared pool of resources and takes its scalability to ten of thousands of nodes.

Spark like Hadoop can be used atop of cloud-based platforms like Amazon Web Services and Google cloud and can utilize the features that utility computing brings in but can do it at a much larger scale with more performance output.

Techniques of data computation and analysis

Expert Systems

An expert system is a system that can imitate the decision-making ability of a human expert in a restricted domain (Giarratano & Riley, 2002). An expert system has a knowledge base that has in it the knowledge base or database of information that makes the expert system an expert in the chosen domain. It also has an inference engine where the analysis happens and a user interface through which the user interacts with the system. The below diagram shows the general structure of an expert system.

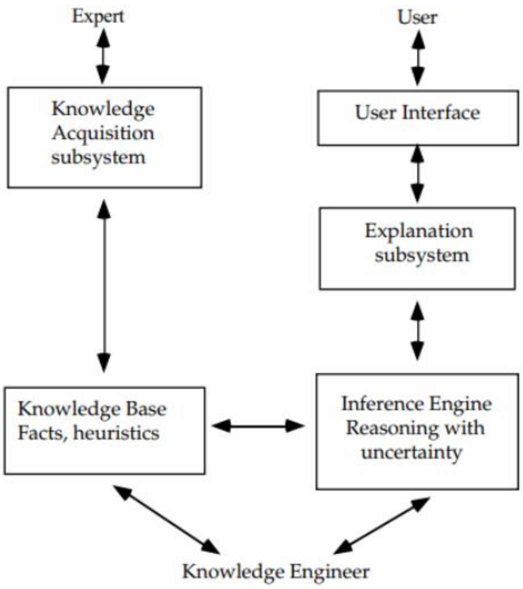
Expert
User
Knowledge Acquisition subsystem
User Interface
Explanation subsystem
Knowledge Base Facts, heuristics
Inference Engine Reasoning with uncertainty
Knowledge Engineer

Fig 5: General Structure of an expert system

An expert system can give a competitive edge to an organization that have integrated it in their operation process as it can analyse large amounts of data at a faster rate and arrive at a decision faster.

Neural networks and Expert Systems

The major application of the neural network is Pattern Recognition. It can recognize a pattern in large amounts of data and come to a conclusion with this data. This is quite like what expert systems do using their knowledge base to come to conclusion in their inference engine. However, the advantage neural networks have over expert systems is that they can update their knowledge base by comparing their inferred results from the dataset with the expected output. Thus, it can be advantageous to develop expert systems using neural networks for data analysis.

The first thing that such a system should do is to identify the problem which in our case could be measuring the performance of a software developer. Then it will have to identify the necessary attribute from the given dataset that is required for doing the analysis and create the training data set. This process is called feature extraction. Using this training data set, the neural network is trained, and it consequently develops a knowledge base.

An expert system as described above can then be used to analyze the predicted performance of a software developer or a software development team based on the indicators given in the dataset which is fed to the neural network during the training process.

Cluster Analysis of Big data

Clustering algorithm can be used for aggregating data pertaining to a company's employees. However, there are numerous clustering algorithms that can be used for doing so. These clustering algorithms can be broadly categorized into the following: -

i. Partitioning – based: - Clusters are formed immediately. Each datapoint in the data set is assigned to only one cluster and each cluster has at least one datapoint. K-means clustering is an example of Partitioning based clustering where the data set is divided into k clusters. For every cluster the centroid is calculated and is optimized iteratively till convergence.

ii. Hierarchical based: - The data is arranged in a hierarchical manner depending upon the dissimilarity of the data-points in the data set. There can be two approaches to hierarchical-based clustering: - agglomerative(bottom-up) or divisive(top-down). BIRCH, CURE and ROCK are some of the popular hierarchical based algorithms.

iii. Density-based: - Data points are separated based on their regions of density, connectivity, and boundary. The advantage of using Density-based clustering algorithm is that it can detect arbitrarily shaped clusters. DBSCAN, OPTICS, DBCLASD and DENCLUE are some of the Density-based algorithm

iv. Grid-based: - The space of the data objects is divided into grids. Wave-Cluster and STING are the well-known Grid-based Clustering algorithm.

v. Model-based: - the data is viewed as being generated from a mixture of underlying probability distributions, each of which is viewed as a different cluster. MCLUST is the best-known algorithm in this category.

Ethical, Moral and Legal Obligations

Measuring the productivity of a software developer is an illusive task. There are several metrics using which the analysis on the developer's productivity can be made and if the metrics the manager chooses to base this analysis upon is something like LoC or commit size, it incorrectly predicts the productivity of that engineer and negatively impacts an organization's efficiency and growth. Moreover, it might be the case that the person ranking the lowest in productivity with this metric is demoted or fired from the organization. This brings upon legal and ethical obligations to the organization along with the moral obligation that's always going to be there if an organization is making use of the developer's personal data to make this analysis.

Therefore, it is much better if the analysis is done at a team's level instead of an individual. Analysing the data on a team's productivity is much less intrusive that an individual's personal data. Moreover, research on this shows that metrics work better if they are aimed at judging the productivity of a team with the correct set of metrics being used by the manager. In this way, much of the ethical, Moral and Legal Obligations can be dealt with.

Conclusion

This report explored the different ways an organization can measure productivity in software development along with the various technologies that can support its development. Overall, it was found that metrics measurement can be very useful to steer the organization towards success by fostering a collaborative and productive approach towards software development when used rightly.

References

1. Singh, D. and Reddy, C.K., 2015. A survey on platforms for big data analytics. *Journal of big data*, *2*(1), pp.1-20

2. A. Fahad et al., "A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis," in IEEE Transactions on Emerging Topics in Computing, vol. 2, no. 3, pp. 267-279, Sept. 2014, doi: 10.1109/TETC.2014.2330519.

3. Ogidan, Ezekiel & Dimililer, Kamil & Kirsal Ever, Yoney. (2018). Machine Learning for Expert Systems in Data Analysis. 1-5. 10.1109/ISMSIT.2018.8567251.

4. Fenton, N.E. and Neil, M., 1999. Software metrics: successes, failures and new directions. *Journal of Systems and Software*, *47*(2-3), pp.149-157.

5. https://aws.amazon.com/emr/features/hadoop/