International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-2, December 2019

Productivity Metric Estimation: Comprehensive Examine of Efficiency, Effectiveness and Value Based Metrics

Harmandeep Kaur, Munish Saini

Abstract: ***The successful software project estimation is critically dependent upon measurement of productivity. The overall goal of productivity metric is to improve the ability and performance of information system evaluation as well as maintenance, moreover determine the performance of the information technology (IT) performance in business applications. The activities corresponding to productivity metrics vary at distinct management levels. The activities used within project management leads to dynamic framework that could be opted in changing or dynamic environments. At the project management level, productivities activities concentrates on effectiveness. For instance, during estimation of effort required to reproduce a system, assumption must be manufacture regarding the ability that could lead to successful development of software project and this is fetched from past projects. This paper focuses on identifying the productivity metrics that are essential and appropriate for project management process.***

Index Terms: Software productivity, ability, effectiveness, efficiency, cost, Information technology.

I. INTRODUCTION

Software productivity is regularly examine as key to achievement, whatever maybe the activity. Singh et al. in2012 [1] discussed applications of package level metric in assessing quality of software product. According to distinct researchers Kaur & Sharma in 2015 [2]and Gupta & Kumar in 2012[3], successful organizations are those that possess packaging quality assessment mechanism and possesses few or all of the following attributes:

- Excellent Performance
- Focus on customer
- Focus on market
- Compressed time
- IT enabled business
- Business Flow
- Cost consumption
- Network in organization
- Orientation and team work
- Strategic and partnership formation
- Organization alliances
- Empowerment of employees

This list is not complete and not ordered but it presents the base for formation of productivity metrics that should be possessed by software project to be effective and efficient.

Revised Manuscript Received on December 30, 2019.
***** Correspondence Author
Harmandeep Kaur*, Department of Computer Engineering and Technology, Guru Nanak Dev University, Amritsar, India.
Munish Saini, Assitant Professor in Department of Computer Engineering and technology, Guru Nanak Dev University,Amritsar ,India.

These metrics are critical for system development and maintenance organization success.
The important question answered in this literature is how to estimate these metrics and how to demonstrate productivity metrics in terms of cost.

A higher level view of project management is concerned with effectiveness of packaging functions delivered to the users by the applications development system. This aspect focuses on effectiveness of delivered product rather than software required to be written to deliver the product. The critical role of management in this aspect is to establish an environment that could be used to maximise the output of required to deliver the product. Thus the use of function point as size metric dependent upon the user view of the system. To provide more effective view of effectiveness of information system function, effort estimation alone could be insufficient and cost metric also should be consider as prime factor influencing the software development process.
The business aspect is greatly dependent upon the contribution of IT while lower level management measure efficiency and effectiveness that are critical for achieving the objectives at project and functional levels. Thus higher level view of software project management is upon value of IT to the business and senior management is concerned with cost and size aspect of the software.

II. REVIEW METHODOLOGY

In review process, consists of the following points:
1) Describing the distinguish research questions. 2) Defining the need of productivity metrics and3) Describing perspectives and issues of productivity metrics.4) Present comparative analyze of metrics evaluation strategies used in distinct literatures.

➢ *Research questions :*

1. Which attributes and techniques have been used for software packaging metric?
2. Is there any evidence of difference in the evolution between the direct metric v/s indirect metric?
3. What are the problems in software productivity metric?
4. How does the preference of using software metrics is changed during the period of time?
5. What is the software productivity explain with evaluation?

➢ The selection criteria :

After a basic selection of 22 papers, the suitable studies were determined subsequently the inclusion and exclusion criteria as follows:

Inclusion- Criteria:

• Include the papers discussing longitudinal analysis of evolution of software productivity metric.

• Include the research studies that combine level of measurement and perspectives of software metric.

• Include studies which propose different strategies to analyze and/or predict software productivity metric.

In other words, this paper reviews studies which focus on the software productivity metric phenomenon. The purpose of this paper is not only to understand the software productivity metric phenomenon per se, but also to learn different strategies that have been employed in perspective of software productivity metric understanding.

Exclusion- Criteria:

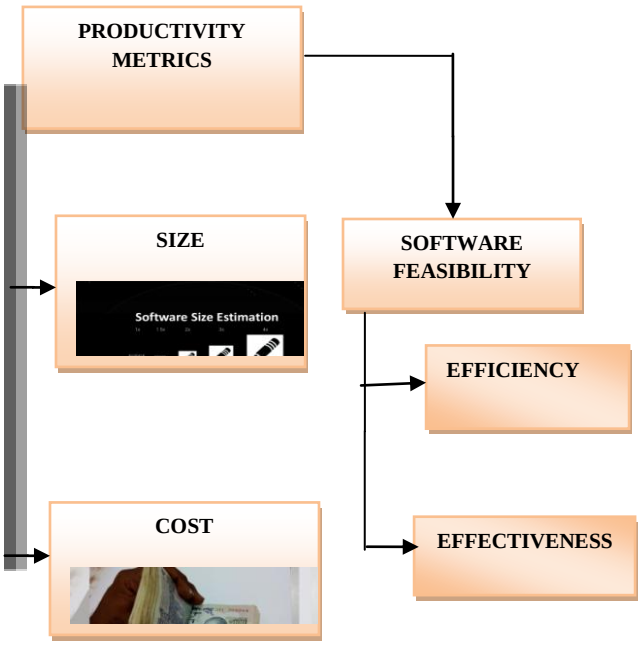
✓ Exclude research studies focusing only on the productivity metrics;

✓ Exclude the studies which don't report any empirical results;

✓ Exclude review strategy.

➢ **Need for package productivity metrics**

J et al. in 2009 [4]proposed value based metric to measure productivity of software. Feasibility greatly dependent upon the requirement gathering process that leads to effective productivity metric development process. Factors such as cost and size greatly impact performance of software development. The cost and size if can be determined in advance than software crises can be avoided. Software feasibility is the prime reason for which productivity metrics are desired in software development process



.

Figure 1: Need of software productivity metrics

Figure 1 elaborates the need for software productivity metrics that could leads to successful feasibility study. The feasibility study if successful than effectiveness and efficiency could be worked upon by lower level management.

III. LEVEL OF MEASUREMENT

A simple perspective of management is to establish continuous chain of planning and control. Snoeck, Poelmans and Dedene, in 2000[5] proposed a metric to measure productivity of programmer. To plan and control, one needs to have guidelines, standards, measures, and measurements. Planning and control is done at different layers of the management process, and one of the common problems is that these distinct layers do not possess the same configuration. Solla et al. in 2011[6]. This work distinguishes three main layers and their different effects on productivity: efficiency, effectiveness, and value. At the executive or top level, one is accustomed to communicating IT execution as far as *'value to the business'*. Cash, conveyance plans, nature of the business procedures, and nature of items and administrations are regular terms in the vocabulary utilized at this dimension. In this specific situation, the utilization of terms, for example, *'powerful lines of code per man-month'* would be equivalent to talking in a foreign language.

Ryan & Connor et al. in 2009 [7] at the higher level of measurement strategically aspect exists. This aspect is formulated so that constructive environment can be built that leads to success of the software development process. For example, in clothe manufacturing industry, strategic decision could be to manufacture woollen clothes in winter and cotton clothes in summers.

Basili et al. 2010 [8]At the intermediate level of measurement, business function management process exists that leads to efficiency and effectiveness corresponding to the software development process. Work corresponding to implementation from the lower level is accomplished by the use of intermediate level.

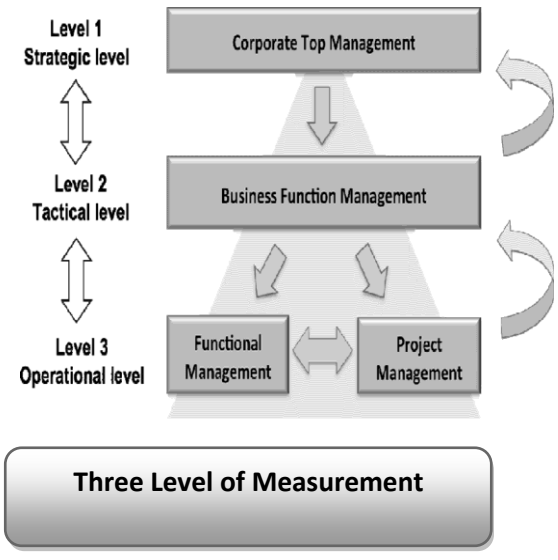


Figure 2: Three level measurement mechanism

Characterize a software development process [9]. The project development level not only establishes guidelines but it also determines cost and eliminates complexity if any associated with the software project.

Time critical application framework development is one of the challenging task in the hands of project management team. These applications must be completed well within time frame specified. This work of determine cost and reducing complexity is accomplished by measurement team at the lower level of management.

At any level of decision making, productivity measurement is concentrated on determining product to effort and value to cost estimation however each level is categorised on the basis of distinct rules and regulations. The discussion of this literature is to effectively determine the mechanism of metric estimation so that measurement could be effective rather than complexity enhancement mechanism.

1. Perspective on Effectiveness and efficiency based mechanisms

The manager at distinct levels of measurements does not have a control over the language and tools used to produce the new system. In demand for meeting the requirements of timeliness, quality and cost the project manager tend to focus on the available resource utilization efficiency. The efficiency could be defined as the total useful work done to the energy consumed [10]. In development process, energy corresponds to effort required for the software development process. The productivity in software development process indicates ratio of value to cost. Effectiveness of software development process is judged in terms of feasibility. Feasibility study at early stage of software development can indicates cost is higher or profit is higher. If cost is higher than software development is denied all together and effort in development is saved greatly.

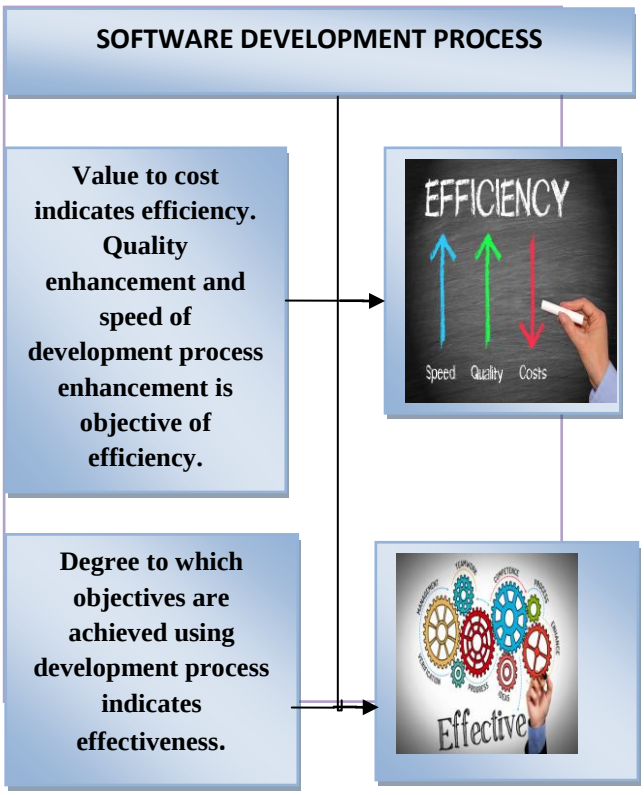


Figure 3: Perspective on effectiveness and efficiency

The perspective on effectiveness and efficiency is generally expressed in terms of line of codes and function point metrics.

1.1 Efficiency Perspective

Frekjmr in 2006 [11]efficiency is expressed in terms of resource utilization. Resource utilization could be low or high. In case of high utilization, necessity of work is identified. To identify necessity of work, useful work done to the total activity is determined. The ratio of efficiency to work must be high.

The ratio of efficiency is defined by the generic definition of productivity is divided by efficient value of useful work performed. This is related to the cost of the productivity and this cost defines the energy expanded. The measurement of useful work is done by energy expanded and with use of energy
perspective.

1.1.1 Measure of size of work using line of code

Suve in 2003 [12]the size of the software is measured in terms of lines of code that is useful for developing the software. This metric takes the number of lines of code that is used to write the program. It is used to estimate the effort made by the user to develop software project. The LOC metric is consistent and it gives exact measure of size of code.

In other words the any line of program that is meaningful for developing project except comment or blank lines are considered in lines of code. The main advantage of using line of code metric is that it is easy to measure and the effort estimation is simple. It mostly used for cost estimation by Boehm model. The effort estimation is done by manager and it utilizes software size for estimation. The estimation is done in terms of man hour required and personnel requirement.

1.1.2 Use of Staff per month to estimate cost

Murray et al. in 2018 [13]the cost is critical parameter determining efficiency of the software product. The efficiency of the software product is at stakes when cost is more than the profit earned. The cost enhances if effort required increases. The effort required is directly proportional to cost since manpower requires payment for operation. The efficiency of the software is impacted in case effort is not accurately determined. For example, if unpaid overtime of worker is not considered then efficiency shown by system is different than it usually will be. Software cost estimation model such as cost constructive model can be employed to estimate effort and cost accurately.

1.2 Effectiveness Perspective

Efficiency is important in success of the software but effectiveness is equally important. Effectiveness estimates the total output delivered by the system that is useful for the user. The effectiveness of the software is given in terms of ratio of useful output to the total output produced by the system. This is also given in terms of value to cost metric.

Murray et al. in 2018 [13]Cost metric cannot be determined in advance by the use of line of code so some other mechanism is desired for predicting it accurately. This is accomplished using function point metric within effectiveness.

1.2.1 Function Point Metric

Xenos & Stavrinoudis et al. in 2000 [14]function point is indirect measure of size and cost associated with the software. Function point metric depends upon internal output, external input, outputs and external variables. Total of 14 distinct environment variables are used for this purpose. The function point metric evaluation characteristics corresponds to table 1:

General System Characteristic		Brief Description
1.	Data Transmission mechanism	How many communication lines are required to transfer the data?
2.	Peer data Evaluation	How data is distributed among multiple systems?
3.	Enhancing the performance of Model	Was performance as desired by user is delivered?
4.	Hardware and software configuration	Was hardware and software used within the users environment is compatible?
5.	Date of transaction	Was frequency of transactions are appropriate?
6.	Data entry subsystem	Does online data entry is supported?
7.	User Conveyance	Does efficiency of work is achieved using the software?
8.	Updating	Does online transaction can be updated?
9.	Processing complications	Does extensive mathematical and logical processing is done using software system?
10.	Usability of resources	Does usability of the software is impressive?
11.	Ease of use	Does software is easy to install in users environment?
12.	Task accommodation	How effective is execution of task through the given system?
13.	Parallel utilization	Does parallel execution of task is supported?
14.	Change accommodation	Were the changes requested by the user can be accommodated within the software?

Table 1: Function Point Metric characteristic table

The function point metric includes indirect analysis of software cost and quality. The quality enhancement is done using function point metric by determining cost and size of the software in advance.

2. Value Based Perspective:

Boehm and Jain, in 2005[15]the third perspective of productivity metric is value based. This metric deals with determination of value for money in the IT infrastructure. Value for money is uncertain metric where even managers are uncertain about this mechanism. Value based perspective deals with the following

- ❖ Related to system development and maintenance cost.
- ❖ Related to cost encounter due to complimentary services
- ❖ Related to investments made due to business requirements.

Value based perspective deals with additional cost encounter due to business needs and hence feasibility is hampered due to additional cost the cost benefit analysis is conducted during value based analysis process.

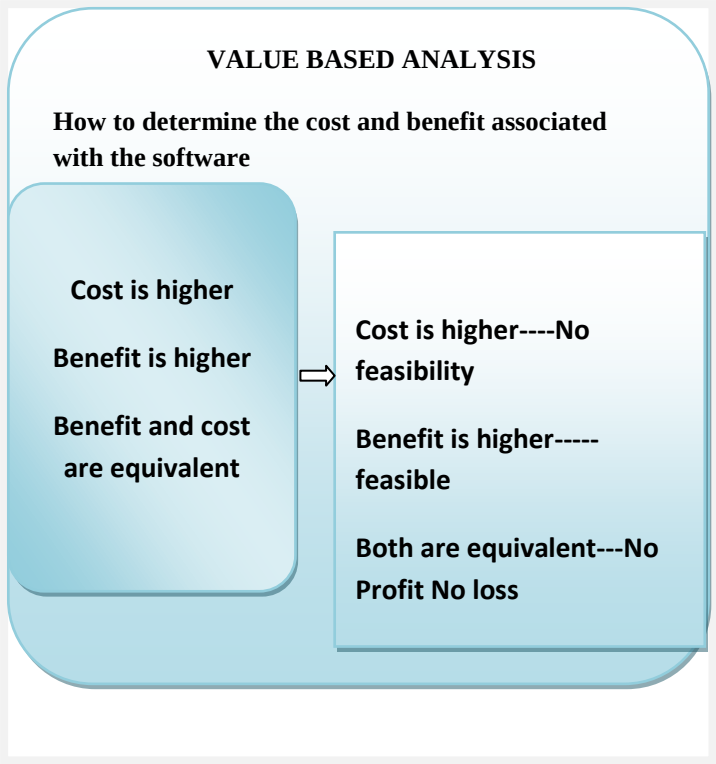


Figure 4: Value based Analysis

IV. ISSUES OF PRODUCTIVITY METRICS

Ahmad, Siddiqui and Khan, in 2018 productivity metrics are necessary to determine the validity of the software product. Estimating productivity of software is a challenging task [16]. The main issues associated with the productivity metric is given as under:

- ✓ Size estimation mechanism varies according to number of line within the software and number of persons served by the software system. This variation causes uncertainty in model development for accurate size determination.
- ✓ Cost estimation depending upon the effort per months.

The indirect measure such as overtime if not included causes uncertainty in development cost. Cost constructive model thus is required to estimate cost that could be misleading if overtime cost is not included.

- ✓ Quality aspect is indirect measure and line of code is direct measure to determine size and cost of the software. LOC however cannot be used in case size and quality is to be determined in the beginning of software development phase.
- ✓ Function point metric estimation mechanism is based on 14 characteristic variables. Effort estimation through this metric is accurate but determining values of these 14 metrics causes cost to increase substantially.

The comparative analysis serves evaluation process used within productivity metric evaluation. The comparative study is given within table2:

V. COMPARATIVE ANALYSIS OF PRODUCTIVITY METRICES USED IN DISTINCT LITERATURE:

Literature	Productivity Metric	Pros	Cons
Hsu & Huang in 2014 [17]	Optimal combination model	Software quality and reliability enhancement	Cost is not considered causing software feasibility to decrease
Singh & Anand in 2017[18]	Domain Dependent Productivity metric	Quality of software is enhanced	Cost and size is not a focused area thus value based analysis fails
Buckley & Buckley in 2017 [19]	Data structure based record detection	Students can use this mechanism to find bugs if any within the software	Cost and execution time is a problem
Rana in 2017 [20]	Required tractability metric	Tractability metric is useful in testing of software	Size and cost determination is an issue
Maragathavalli & Saranya in 2017 [21]	Adaptive random testing	Bug discovery is quick	Size and cost adaptability is missing
Fischbach, Wiebusch and Latoschik, in 2017 [22]	Semantic based state model	Modularity, modifiability and reusability is ensured using this model	Size is reduced but cost is increased

Table 2: Comparative analysis of metrics used within distinct literatures

Concluding part from the comparative analysis is least amount of work is done to enhance size and cost of the software. In future, software model can be built for determining optimal size of the software by eliminating unnecessary modules such as documentation module could lead to decay in size and ultimately cost is decreased.

VI. RELATED WORK

In the literature discussed above different strategies associated with productivity metrics are discussed and evaluated. The approach of productivity metric determines best possible mechanism for code formation. The code formation must be structured in nature following single entry single exit constructs. The constricts may be graphical or command user in nature. The graphical user interface is easy to understand and evaluate and in most of the literature GUI environment is used for code evaluation.

Quality of code segment is defined in terms of number of requirements satisfied by the software module. Reliability of code indicates the total satisfied requirements to the total requirements given by user in software requirement specification document[23].Quality of code segment depends upon the testing phase and is partitioned into four different parts. First parts deals with the requirement analysis, second phase consist of checking the feasibility of code segment or in other words checking weather required material can easily be managed or not. Third phase is associated with design code segment and last phase involves implementations and testing.

Code quality analysis mechanism is accommodated within the analysis phase to determine validation of requirements. Validation phase includes checking requirements against the test module. Nowadays software's such as sonarqube is used to determine quality of software product. Code modules generally contains bugs at the interface section. To check for the bugs at the interface section, unit testing can be accommodated within the test module. Test phase is negative and its success depends greatly on the bugs determined to improve quality of code segment.

Trendowicz*et al.*[24]discussed the factors that are going to influence quality of code segment. Quality of code segment greatly depends upon metric evaluation mechanisms and SRS satisfaction criteria's. There are tools that could influence overall operation of quality checking mechanism. Code and design quality determines final product that is formed and delivered to the user. It's not only the quality at the development stage but after delivery maintenance quality that is going to determine quality [25].

Productivity of software in packaging can be influenced by data based approach used in the discussed literature.

Coupling and cohesion mechanism plays critical role in determination of necessary code segments. Unnecessary code modules can be skipped to form structured code segments for enhancing code quality[26].

Discussed integration of data and structured based approach for enhancing code quality[27].there exists tools that could be used to form code segment and enhancing quality of software. These tools are generally structured in nature. This mechanism includes data dictionary, data flow diagrams etc. each of these modules has different types of interfaces. These interfaces includes command based and graphical both.

De Aquino and De LemosMeira[28], discussed SLOC and function point metric to determine effective size of software. The function point metric is indirect measure and line of code is direct measure to determine size of the software. Line of codes and size directly reflects the cost associated with the software. Size and cost minimization is the objective of this study. in [29], discussed effort required to form a code segment. Effort is expressed in the form of person per month. To find effort required socio economic factors must be determined accurately.

Functional user requirement is building block determining successful software development and effective quality satisfying user requirements. Functional requirement leads to formation of modules and top down approach is followed to satisfy requirements of the users [30].

Type and usage attributes are discussed through this literature that is essential in packaging software. Metric required to be considered includes cost and size. Both the metrics are directly proportional to each other [31].

In proposed Model in [32], discussed three levels of management that must be provided with the metrics support for the development of software module. Every level has different software requirements along with the effort requirement. The common model that is used for the purpose of detection of cost is cost constructive model. This model gives the cost of the code module in advance to determine cost and benefit analysis. If cost is high and benefit is low then software module is rejected otherwise software module is created and passed to the user for evaluation.

An exploration about the estimation utilizing the procedure of Genetic Programming (GP) for investigating preferable cost functions. At that point analysts additionally included non-parametric demonstrating methods in light of machine learning calculations and similarity in the relative examination. It is contrasted a similarity based method and stepwise relapse. They utilized nine unique informational collections from various areas and report that, in all cases similarity outflanks stepwise relapse models as far as the MMRE. The paper utilized Kemmerer's task informational index and found that their relationship based model Estor, utilizing case-based reasoning (CBR), beat the COCOMO mode [33].

It contrasted CBR and diverse relapse models utilizing FP and simulated neural systems on a vast database comprising of 299 undertakings from 17 unique associations. They report a superior execution of CBR when contrasted and distinctive relapse models in view of function points. What's more, counterfeit neural systems outflanked the CBR approach. It included relapse trees, manufactured neural systems, function points, the COCOMO display, and the SLIM model in their correlation. They utilized the COCOMO informational collection (63 ventures from various applications) as a preparation set and tried the outcomes on the Kemerer information (15 ventures, for the most part business applications). The relapse trees beat the COCOMO and the SLIM model. They likewise found that counterfeit neural systems and function point based forecast models beat relapse trees. Utilizing a mix of the' COCOMO and the Kemmerer informational collections, Briand [34].

Jorgensen utilized 100 support projects for testing a few varieties of relapse, counterfeit neural systems, and blends of OSR with relapse. He found that two numerous relapse models and a mixture show joining OSR with relapse worked best as far as exactness. All in all, he prescribed the utilization of more advanced forecast models like OSR together with master evaluations to legitimize the interests in those models.

Although parametric procedures are incorporated into the majority of the investigations looking at changed cost estimation strategies, the examinations are halfway as in just certain systems are assessed. Besides, replications of studies are once in a while performed. Even when similar informational index is utilized as a part of various examinations, the outcomes are not generally practically identical due to various trial plans. For instance, both utilized the COCOMO and Kemmerer information; nonetheless, they utilized the information in various routes as preparing and test sets.

Besides, numerous examinations utilized just little informational collections originating from various situations. This makes it hard to make generalizable determinations about the models' execution [35].

The analysis of the literature [36] indicates that the broader aspect associated with the software productivity must be considered rather than smaller perspective. Cost effective mechanism and models discussed such as cost constrictive model and metrics such as function point allow to check the quality of software.

In addition software packaging suffer from crisis due to high rejection rate. To check for the success of software packaging and increase productivity, parallel conversion mechanism is used. The parallel conversion indicates usage of old as well as new mechanism simultaneously.

Association rules helps in determining productivity of code module in testing modules. .These rules can be formed using fuzzy based interface engine. New rules allow wide coverage of code segment and hence code quality can improve. New rules can also ensure better formation of framework to check for the code quality [37].

VII. RESULT AND DISCUSSION

There we analysis the various techniques that are used to measure software productivity. The table below shows the results :

LOC	Statements	Classes	Functions
<1000	3078	70	287
>1001	2467	58	240
<1500 >1501	2978	244	2078
64567	42345	378	3456

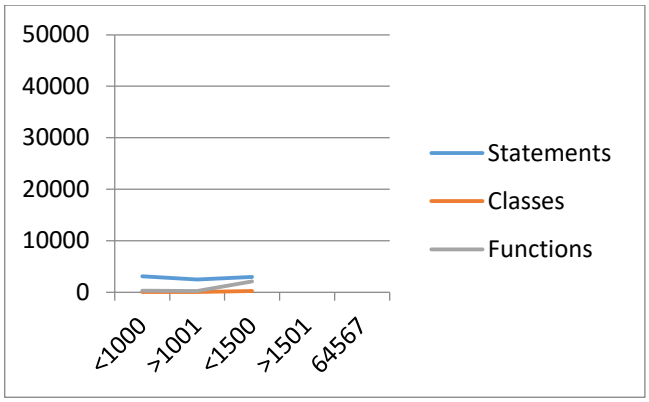


Figure: Comparison of various metrics used for software productivity

VIII. CONCLUSION

Software development is critical in determination of cost and effort and reducing software crisis. The quality is also an issue that must be enhanced. To accomplish the same software development productivity metrics must be considered. The productivity metrics are much worked upon by the literatures but cost and size is not prime objectives in either of the consider literatures. Size and cost could be reduce by considering cost constructive model through which effort estimation could be effectively estimated. The cost estimates if accurate could lead to successful software development process.

IX. FUTURE WORK

In future, size reduction using documentation module elimination could be proposed along with cost reduction using overtime cost reduction mechanism.

REFERENCES

1. V. Singh, "Evaluation and Application of Package Level Metrics in Assessing Software Quality," IJCA, vol. 58, no. 21, pp. 38–46, 2012.
2. G. Kaur and D. Sharma, "A Study on Robert C . Martin ' s Metrics for Packet Categorization Using Fuzzy Logic," IJHIT, vol. 8, no. 12, pp. 215–224, 2015.
3. V. Gupta and J. Kumar, "Package level cohesion measurement in object-oriented software," springer, pp. 251–266, 2012.
4. [4] G. S. D. A. J, S. Romero, and D. L. Meira, "An Approach to Measure Value-Based Productivity in Software Projects," IEEE Access, 2009.
5. M. Snoeck, S. Poelmans, and G. Dedene, "Conceptual Modeling — ER 2000," vol. 1920, no. October, 2000.
6. M. Solla, A. Patel, and C. Wills, "New Metric for Measuring Programmer Productivity," IEEE Conf., pp. 177–182, 2011.
7. S. Ryan and R. V. O. Connor, "The Journal of Systems and Software Development of a team measure for tacit knowledge in software development teams," J. Syst. Softw., vol. 82, no. 2, pp. 229–240, 2009.
8. V. R. Basili et al., "and Business Measurement," IEEE Access, no. April, pp. 57–65, 2010.
9. M. A. Akbar et al., "Improving the quality of software development process by introducing a new methodology-Az-model," IEEE Access, vol. 6, no. February, pp. 4811–4823, 2017.
10. M. Z. Khan, A. Alsaeedi, and M. Huda, "Empirically Validated Software Efficiency Estimation Model: Product Operation Perspective," J. Softw. Eng. Appl., vol. 11, no. 10, pp. 486–499, 2018.
11. E. Frekjmr, "Measuring Usability : Are Effectiveness , Efficiency , and Satisfaction Really Correlated ?," IEEE Access, vol. 2, no. 1, pp. 345–352, 2006.
12. D. Suve, "JAsCo : an Aspect-Oriented approach tailored for Component Based Software Development," IEEE Access, pp. 21–29, 2003.
13. N. Murray, G. Muntean, S. Member, Y. Qiao, S. Brennan, and B. Lee, "Modeling User Quality of Experience of Olfaction-Enhanced Multimedia," IEEE Access, pp. 1–13, 2018.
14. M. Xenos and D. Stavrinoudis, "OBJECT-ORIENTED METRICS – A SURVEY," IEEE Access, no. November 2013, 2000.
15. B. Boehm and A. Jain, "An Initial Theory of Value-Based Software Engineering BT - Value-Based Software Engineering," Value-Based Softw. Eng., no. January 2005, pp. 1–30, 2005.
16. A. Ahmad, T. Siddiqui, and N. A. Khan, "A Detailed Phasewise Study on Software Metrics : A Systematic Literature Review," Int. J. Sci. Res. Comput. PSci. Eng. Inf. Technol., vol. 3, no. 3, pp. 1696–1705, 2018.
17. C. Hsu and C. Huang, "Optimal Weighted Combinational Models for Software Reliability Estimation and Analysis," IEEE, vol. 63, no. 3, pp. 731–749, 2014.
18. O. Singh and A. Anand, "Testing Domain Dependent Software Reliability Growth Models," IEEE Access, vol. 2, no. 3, pp. 140–149, 2017.
19. I. A. Buckley and W. S. Buckley, "Teaching Software Testing using Data Structures," Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 4, pp. 1–4, 2017.
20. A. Rana, "Process of finding defects in software testing," IEEE Access, vol. 10, no. vi, pp. 297–300, 2017.
21. P. Maragathavalli and A. Saranya, "Prioritization on Software Reliability Assessment using Adaptive Testing Strategy," springer, vol. 2, no. 3, pp. 273–277, 2017.
22. M. Fischbach, D. Wiebusch, and M. E. Latoschik, "Semantic Entity-Component State Management Techniques to Enhance Software Quality for Multimodal VR-Systems," IEEE Access, vol. 2626, no. c, 2017.
23. D. Contreras and M. Castro, "Adaptive polling enhances quality and energy saving for multimedia over bluetooth," IEEE Commun. Lett., vol. 15, no. 5, pp. 521–523, 2011.
24. M. Fischbach, D. Wiebusch, and M. E. Latoschik, "Semantic Entity-Component State Management Techniques to Enhance Software Quality for Multimodal VR-Systems," IEEE Trans. Vis. Comput. Graph., vol. 23, no. 4, pp. 1407–1416, 2017.
25. H. Liu, "Software Engineering Practice in an Undergraduate Compiler Course," IEEE Trans. Educ., vol. 36, no. 1, pp. 104–108, 1993.
26. D. I. Heimann, "IEEE Standard 730-2014 Software Quality Assurance Processes Learning objectives What is IEEE 730 ?," pp. 1–26, 2015.
27. C. J. Hsu and C. Y. Huang, "Optimal weighted combinational models for software reliability estimation and analysis," IEEE Trans. Reliab., vol. 63, no. 3, pp. 731–749, 2014.
28. P. M. S. S. Chandu and D. Kata, "Integrating and enhancing the quality of services in cloud computing with software testing," Proc. 2016 IEEE Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2016, pp. 2008–2010, 2016.
29. Deepika, O. Singh, A. Anand, and J. N. P. Singh, "Testing Domain Dependent Software Reliability Growth Models," Int. J. Math. Eng. Manag. Sci., vol. 2, no. 3, pp. 140–149, 2019.
30. W. Wang, P. Xu, X. Bie, and M. Hua, "Enhanced Use of Mattes for Easy Image Composition," IEEE Trans. Image Process., vol. 25, no. 10, pp. 4608–4616, 2016.
31. A. Rahman, F. H. Sunny, H. M. Mishu, and F. Sumi, "Open Access Software Testing Algorithm Units," no. 1, pp. 271–275, 2017.

32. S. K. Sharma, “Reliability Growth Models for Software Testing Against Security Requirements,” vol. 6, no. 6, pp. 126–132, 2017.
33. K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, “Data Mining Techniques for Software Effort Estimation : A Comparative Study,” vol. 38, no. 2, pp. 375–397, 2012.
34. E. Kocaguneli, S. Member, T. Menzies, J. Keung, D. Cok, and R. Madachy, “Active Learning and Effort Estimation : Finding the Essential Content of Software Effort Estimation Data,” pp. 1–14, 2012.
35. T. W. R. L. K. C. Kang, “Effort estimation of component-based software development – a survey,” vol. 5, no. June 2009, pp. 216–228, 2011.
36. J. F. Nunamaker and M. Chen, “Software Productivity: A Framework of Study and an Approach to Reusable Components,” IEEE Access, 1989.
37. N. Murray, G. M. Muntean, Y. Qiao, S. Brennan, and B. Lee, “Modeling user quality of experience of olfaction-enhanced multimedia,” IEEE Trans. Broadcast., vol. 64, no. 2, pp. 539–551, 2018.

Author profile

HARMANDEEP KAUR
Department of Computer Engineering and Technology,Guru Nanak Dev University, Amritsar, India.Harmandeep Kaur is pursuing Mtech (CSE) from Guru Nanak Dev University, Amritsar. Her research interests include software engineering and open source software.She did her Bachelors of Engineering in Computer Science and Technology from Global Institute of Management and Emerging Technologies, Amritsar, India.

MUNISH SAINI
Munish Saini is an Assitant Professor in Department of Computer Engineering and technology,Guru Nanak Dev University,Amritsar ,India. He received his phD in computer Science and Engineering from Dr. B.R Ambedkar National Institute of Technology. Jalandhar,India . His research Interest Is in Data Mining, Open Source Software,and Software Engineering.