Efficiency Metrics

Tamanna Siddiqui[1], Munior Ahmad Wani[2] and **Najeeb Ahmad Khan[3]**

Submitted in May 2011, Accepted in June 2011

Abstract - Software measurement is a challenging but essential component of a healthy and highly capable software engineering culture. It is an integral part of the state-of the-practice in software engineering. More and more customers are specifying software and/or quality metrics reporting as part of their contractual requirements. Software Engineering has always been a matter of concern for every individual involved in software development starting from analysis phase to delivery phase or even at the maintenance time. There have been novel approaches for developing program complexity metrics. In this regard we have proposed the Efficiency Metrics, which can calculate the efficiency of a programmer and can also calculate the exact time taken by the development team to complete the software development under various complexities. Over and above we have also developed a relation between time and efficiency.

Index Terms - LOC, Mean, Standard Deviation, Low, Medium, High, Errors, Delay Time, Committed Time

1. INTRODUCTION

There have been novel approaches for developing program complexity metrics. The first which was developed by Halstead[16], uses a series of software science equations to measure the complexity of a program. McCabe[17], uses graph theoretic measures to define a cyclomatic complexity metric. Albretch[18], who hypothesized that the amount of function to be provided by an application program can be estimated form an itemization of the major components of data to be used or provided by it. In this regard we have proposed the Efficiency Metrics, which can calculate the efficiency of a programmer and can also calculate the exact time taken by the development team to complete the software development under various complexities. Fear is often a software practitioner's first reaction to a new metrics program. People are afraid the data will be used against them, that it will take too much time to collect and analyze the data, or that the team will fixate on getting the numbers right rather than on building good software [20]. Creating a software measurement culture and overcoming such resistance will take diligent, congruent steering by managers who are committed to measurement and sensitive to these concerns. Software metrics, presented in various textbooks, e.g. [11],[12],[13],[14] and conferences and workshops [12], has a long tradition in theory, while considerably shorter in terms of industrial applications. Software metrics relies on the underlying theory, called representational measurement theory, posing some requirements on a correct definition, validation, and use of software metrics. From practical point of view, there are several further questions of importance, e.g. how to identify the right metrics to use, how to introduce a metrics programme, and how to keep it alive. Software process and product metrics are quantitative measures that enable software people to gain insight into the efficacy of software process and the projects that are conducted using the process as a framework. Basic Quality and productivity data is collected. This data is then analyzed, compared against the past averages, and assessed to determine whether quality and Productivity improvements have occurred or not [7]. Metrics are also used to pinpoint problem areas so that remedies can be developed and the software process can be improved [5].

A comparison of software metrics by Halstead, McCabe and Albrecht, in terms of their ability to measure software productivity has led to the conclusion that in the areas where it is applicable, the function point metric is the best of the three[14]. It should be noted that the values of Halsted's metrics becomes available only after the coding is done and therefore can be of use only during the testing and maintenance phase. The increasing demand of the software industry across the globe is that it needs both the development of improved software metrics and improved utilization of such metrics.[1] Software metrics can be classified into product metrics & Process Metrics or Objective Metrics & Subjective Metrics. On these bases many Software Models and Software Metrics have been proposed like Size Metrics by Boehm & Johns [8], Function point Metrics by Albrecht, Bang Metrics by Demark, Information Flow Metrics by Kafure & Henry etc. Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined unambiguous rules. A good measurement program is an investment in success by facilitating early detection of problems, and by providing quantitative clarification of critical development issues. Metrics give you the ability to identify, resolve, and/or curtail risk issues before they surface. Measurement must not be a goal in itself. It must be integrated into the total software life cycle — not independent of it [10]. Different type of measurement for different parameters of software product is possible through different types of metrics. Proposed research work is an effort to present a Delay metrics, which will solve the problem of time delay in software development.

[1,2,3]Department of Computer Sciences. Jamia Hamdard, Hamdard University, New Delhi, India
E-mail: [1]tsiddiqui@jamiahamdard.ac.in,
[2]muneer.wani@gmail.com and [3]nakhan@jamiahamdard.ac.in

2. RELATED WORK

Many Researchers have been working on the exact time of development and they have also succeeded to some extent. Halstead and Raleigh has been able to find the development time however the results would have been more accurate, had the efficiency of the programmers also been taken into consideration.[1][2] Goal-question-metric (GQM) is an effective technique for reducing the average time and to close a defect by 40 percent within three months. However it too lacks the programmers efficiency in its calculations [4], because the distribution of reasons for delay varied widely from one department to another, it is recommended that every department should gain an insight into its reasons for delay in order to be able to take adequate actions for improvement [2]. The field of software engineering especially in the field of software metrics the success rate is not that good because most of the software development companies avoid to follow the proposed metrics. Project initiation is a good time to choose the appropriate measures that will help developer to assess project performance and product quality [6]. To plan measurement activities carefully will take significant initial effort to implementation and the payoff will come over time [3].

Yin and Winchesters Metrics [15], which depend on design structure can be useful in identifying sections of a design that may cause problems during coding, debugging, integration and modification. This metrics is available from the design phase onwards and hence can be used to predict values like the number of errors in the system, time for system testing, time for rectification of errors etc. Henry and Kafura's Metrics[15] is an appropriate and practical basis for measuring large scale systems. The major elements in the information flow analysis can be directly determined at design time, thereby allowing any corrections in the system structure with the minimum cost. Also by observing the patterns of communication among system components, it is possible to define measurements for complexity, module coupling, level interactions and stress points in the design. These critical system qualities cannot be derived from simple lexical measures. In a nutshell we can say that this metric is to determine the complexity of a procedure which depends on two factors: the complexity of the procedure code and the complexity of the procedures connections to its environment[15]. Once the errors are predicted by Yin and Winchester Metrics and the complexity of the code calculated by Henry and Kafura;s metrics, there is a need to develop a metrics which will calculate the exact time of development being the complexity of a procedure or program its important parameter[19].

Background of the early depicted Software Models:

2.1 COCOMO Model

The most fundamental calculation in the COCOMO model is the use of the Effort Equation to estimate the number of Person-Months required to develop a project. Most of the other COCOMO results, including the estimates for Requirements and Maintenance, are derived from this quantity. The original COCOMO 81 model was defined in terms of Delivered Source Instructions, which are very similar to SLOC. The major difference between DSI and SLOC is that a single Source Line of Code may be several physical lines. For example, an "if-then-else" statement would be counted as one SLOC, but might be counted as several DSI. However the efficiency of the programmer is not taken into consideration while performing such calculations to meet the deadlines of the client.

2.2 Waterfall model

The waterfall model however is argued by many to be a bad idea in practice, mainly because of their belief that it is impossible to get one phase of a software product's lifecycle "perfected" before moving on to the next phases and learning from them. A typical problem is when requirements change midway through, resulting in a lot of time and effort being invalidated due to the "Big Design Up Front". Only a certain number of team members will be qualified for each phase, which can lead at times to some team members being inactive. Had the programmers efficiency been checked before handing them over this job, the project manager could have assigned high efficiency programmers for coding.

2.3 Spiral model

In spiral model the software is developed in a series of incremental releases with the early stages being either paper models or prototypes. Later iterations become increasingly more complete versions of the product. Major flaws identified in spiral model is that Demands considerable risk-assessment expertise and has not been employed as much proven models .

2.4 Java Execution model

Though this model can check the performance of the software developed in Java but still lacks the time and efficiency constraints.[21]

Any of these COCOMO, WaterFall or Spiral models have been run in the software industry but when there are sharp deadlines for the completion of the project by client, such models become obsolete without housing the efficiency metrics.

2.5 Relation with Defect Removal Efficiency

Defect Removal Efficiency (DRE) is a measure of the efficacy of your SQA activities.. For eg. If the DRE is low during analysis and design, it means you should spend time improving the way you conduct formal technical reviews.

$DRE = E / (E + D)$

Where E = No. of Errors found before delivery of the software and D = No. of Errors found after delivery of the software. [22]

Remedy: If DRE is low during analysis and design, we could find the efficiency of programmers and put the best ones for coding purposes to meet the deadlines of client in time bound and result oriented fashion.\

2.6 Feature Performance Metrics

Firstly, relative value is measured by the impact that each feature has on customer acquisition and retention. Secondly, feature value is compared to feature cost and specifically development investment to determine feature profitability. Thirdly, feature sensitivity is measured. Feature sensitivity is defined as the effect a fixed amount of development investment has on value in a given time. Fourthly, features are segmented according to their location relative to the value to cost trend line into: most valuable features, outperforming, underperforming and fledglings. Finally, results are analyzed to determine future action.[23]

3. PROPOSED WORK

If there are twenty programmers hired by the company, though there language skills, technical knowledge and aptitude is checked by the recruitment team, however it is not necessary that all of them would be having same expertise in a particular programming language or their level of aptitude and typing skills. So it is necessary to check their efficiency before assigning them the projects. Based on the efficiency, the work force management team of the organization shall assign the programmer a particular module of development where he/she can give their best with less assistance. If we don't measure our current performance and use the data to improve our future work estimates, those estimates will just be guesses. Because today's current data becomes tomorrow's historical data.

We have tried this efficiency metrics at the initial phase of the software, after analysis. The team leader (project in charge) took up the manpower for his assigned project, based on this efficiency metric. He picked up the people whose efficiency rated (7-9) for very complex modules, (4-6) for normal modules and (3-4) for easy modules, be it designing or coding.

In this paper we propose efficiency metric in which we are using three constants:

Programmers Status	1	Fresher
	2	Intermediate
	3	Experienced
Function complexity	1	Low
	2	Medium
	3	High
Efficiency Constant	100	% calculator

The proposed efficiency metric is defined as:

$$E_{(Prog)} = \sqrt{\frac{F_{(c)} x LOC_{(d)} x e}{P_{(s)} \times T_{(c)}}}$$

Where,

$E_{(Prog)}$ is the efficiency of a programmer in a project.
$F_{(c)}$ is the function complexity
$LOC_{(d)}$ is the lines of code developed for assigned function.
$P_{(S)}$ is the programmer's status.
$T_{(c)}$ is the total time consumed (in minutes) for developing the Lines of code.
e is an efficiency constant and its value is 100.

4. EXPERIMENT

In Table 1, value of fifth column is the value of efficiency of programmer, which is obtained by the proposed metric.

Programmer Status	F(c)	LOC (d)	T(c)	E (prog)
1	1	5	3	3
2	1	5	2	4
3	1	5	1	8
1	2	7	7	2
2	2	7	6	3
3	2	6	3	6
1	3	9	13	2
2	3	8	11	3
3	3	8	8	4

Table 1: A Table (Sample Data) Calculating the Efficiency of a Programmer for a Software Development Project

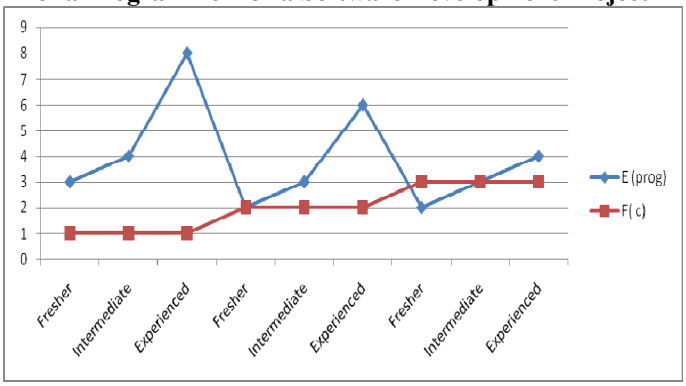

Figure1: Efficiency Graph

Figure 1 shows the efficiency of different programmers at development of functions of different complexities.

By having a look at the chart 1 above, it is clear that the efficiency of programmers do not vary much when we need to

develop programs of simple complexity however there is much difference once we go on higher complexity levels. Higher complexity level projects demand more experienced programmers for the completion within the stipulated time period.

Meeting the committed deadlines before testing the manpower with efficiency metrics.			**Meeting the committed deadlines after testing the manpower with efficiency metrics.**	
Committed deadline after analysis	75 days		Committed deadline after analysis	75 days
Deviation from committed time	22 days		Deviation from committed time	6 days

A relation between time taken and efficiency:
We have analyzed the data of Oriole InfoTech (A software company of repute) as depicted in Table 2, where programmers of any status are given the suite to develop, and we have found that as time taken for development of code is more, the efficiency of the programmer is less. (Table -2) is an extraction of the two parameters Time and Efficiency from Table -1

T(c)	E (prog)
3	3
2	4
1	8
7	2
6	3
3	6
13	2
11	3
8	4

Table 2

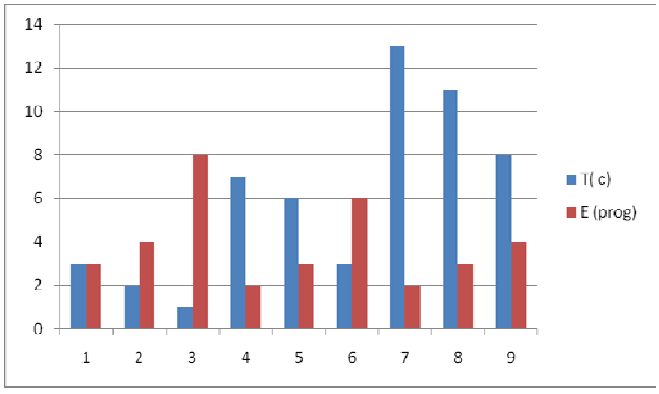

Figure 2: Time and Efficiency Graph

From the above Table-2, we have depicted the following bar graph which clearly states that the efficiency of a programmer is inversely proportional to time.

$$E \propto \frac{1}{T}$$

Where,
T(c) is the time consumed in development and
E(Prog) is the efficiency of the programmer.

5. CONCLUSION

Though the changes in the analysis, design and code are certain, we can still calculate the efficiency of the manpower (Programmers) before we involve them in a project of development. We shall be able to reap better results by assessing the past development data from knowledge bases of various companies and learn by the development hurdles which they have faced. The programmer's efficiency table shall be able to calculate the efficiency of the programmer to an appropriate level based on his a+ptitude, typing and programming skills. This efficiency shall allow us to forecast the manpower required for development of a project under certain level of complexity, to be very close to the deadline of the client.

FUTURE SCOPE

Since Yin and Winchester metrics plays a vital role in the design phase of software development, Henry and Kafura's metrics serves as a base for our efficiency metrics as it helps us to access the complexity of a procedure. Both these metrics are helpful till the design phase however become obsolete when we enter the coding domain of software development. So our efficiency metrics will help us to a great extent in the coding part of the software development process. However the proposed software metrics is rarely followed by the companies of repute because of the reasons best known to them [6]. So it would be better if all the software companies of repute tie up with good academic institutions so that the researchers get the exact past development data to come up with an appropriate knowledge base which will help us to make future software metrics to maintain and manage domestic and global deadlines.

REFERENCES

[1]. SEI Curriculum Module SEI-CM-12-1.December 1988 Everald E. Mills Seattle University
[2]. IEEE Transactions on Software Engineering, Volume 17 , Issue 6 (June 1991), Pages: 582 - 590 Year of Publication: 1991 ISSN:0098-5589 Author Michiel van Genuchten
[3]. C12625211.fm Page 153 Monday, July 9, 2007
[4]. Daskalantonakis 1992; Basili and Rombach 1988
[5]. D. Radoiu, A. Vajda, Department of Mathematics and Computer Science Petru Maior University,Tirgu Mures

Romania. STUDIA UNIV. BABES BOLYAI, INFORMATICA, Volume XLIX, No. 2, 2004

[6]. Cem Kaner, Senior Member, IEEE, and Walter P. Bond, 10TH INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, METRICS 2004 KANER / BOND - 1

[7]. R. S. Pressman, *Software Engineering: A Practitioners Approach* (McGraw Hill, NY, 2005).

[8]. B. Boehm, *Software Engineering Economics*. (Prentice-Hall, Englewood Cliffs, NJ, 1981).

[9]. 12 Steps to Useful Software Metrics, Linda Westfall The Westfall Team westfall@idt.net PMB 101, 3000 Custer Road, Suite 270 Plano, TX 75075

[10]. Book: Software Estimation, Measurement & Metrics GSAM Version 3.0 Fenton, N. Whitty, R., Iizuka, Y., Software Quality Assurance and easurement. A Worldwid Perspective. International Thomson Computer Press, 1995.

[11]. Fenton N: Software Metrics for SPI, Workshop on Process Improvement, Eurex, London, January 1999

[12]. Humphrey W: A Discipline for Software Engineering, Addison-Wesley Publishing Company 1997.

[13]. Paulk, M.C. et al, The Capability Maturity Model Guidelines for Improving the Software Process, Addison-Wesley, 1995.

[14]. A survey of system complexity metrics, J.K Navlakha, Department of mathematical Sciences, Florida International University, Miami, Florida 31399,U.S.A.

[15]. M.H. Halstead, Elements of software science, Elsevier, Amsterdam (1977).

[16]. T.J. McCabe, A complexity measure IEEE, Transactions on software engineering, SE-2 (4), 308 – 320.

[17]. A.J. Albrecht and J.E. Gaffney, Jr, Software function source lines of code, and development effort prediction: a software science validation, IEEE Transactions on software engineering SE-9 (6), 639 – 648 (1983).

[18]. The computer Journal, Volume 30, No 3, 1987.

[19]. A software metrics, C12625211.fm Page 153 Monday, July 9, 2007

[20]. Important aspects in Load & Performance Testing – 3 – Steps and things to test in a component for performance (2010)

[21]. Software Quality Metrics, Parent Category: Software Testing Major Editors: R K akshaya bhatia stevetuf (2010)

[22]. Feature Performance Metrics for Software as a Service Offering: The Case of HubSpot Avi Latner MIT, System Design and Management Cambridge, MA, USA Ricardo Valerdi MIT, Lean Advancement Initiative Cambridge, MA, USA (2011)