ISSN 2047-3338

Proposed C.E.M (Cost Estimation Metrics): Estimation of Cost of Quality in Software Testing

Latika Kharb
Faculty of IT (MCA), Jagan Institute of Management Studies (JIMS), Sector-5, Rohini, Delhi, India
latika.kharb@jimsindia.org

***Abstract*– Reliability in software is produced by controlling its quality within budget. Over the time, researchers and practitioners have expressed their inability to accurately estimate costs associated with software development and it has become even more problematic as costs associated with development continue to increase. As a result, a lot of research attention is now directed towards gaining a better understanding of the software development process as well as evaluating software metrics for cost estimation in the process of software testing. An effective software testing cost measurement and/or evaluation technique requires a metrics that could not only increase the reliability, reusability, correctness and maintainability but also measures the quality and productivity of software development process within budget. Keeping this goal in mind, we've provided a set of three useful Cost Estimation Metrics (CEM) that could be effectively implemented to evaluate costs of testing the software products, for improvement in overall software product reliability, and also become one of the metrics used for the measurement of overall expenses incurred during the process of software testing and thus make way towards a reliable and quality oriented software system in recent future.**

***Index Terms*– Cost Estimation Metrics, Quality Improvements, Quality Software System, Quality within Budget and Quality Oriented Software System**

I. INTRODUCTION

SOFTWARE projects are illustrious for going beyond their deadline, going over budget, or both and the problem lies in the estimation of the amount of effort required for the development of a project. In early days of computing, software costs constituted a small percentage of overall computer-based system costs but today; software is the most expensive element of virtually all computer based systems. Software cost estimation is a tough job, as a number of factors can affect ultimate cost of software to develop it. For large and complex systems, a small cost -estimation error can become tragic for the software developer. As cost estimation is the approximate judgment of the costs for a project [1] i.e., a set of techniques and procedures that is used to derive the software cost estimate; and the purpose of software cost estimation is to:

- Define the resources needed to produce, verify, and validate the software product, and manage these activities.
- Quantify the uncertainty and risk inherent in this estimate.

Even today, much of the metrics activity in industrial sector is based on metrics invented long ago in 70's. This mismatch exists between them because of the following reasons [2]:

- Researchers/academicians are mainly concerned with detailed and code-oriented metrics while the industrial sectors demand those metrics that could help them in their software process improvement. This difference between needs is the main cause behind the mismatch between usability criteria of various metrics.
- Industrial sectors have to abide by some rules and regulations/standards of their company while academicians/researchers are not bound by any such rigid standards and can select/change metrics when ever needed according to their needs and requirements.
- Researchers go for relatively small field works with small data (consisting of small programs) that could get them quick outputs. But the industry men have to go for large projects (to develop huge software). In academics, metrics may or may not be evaluated for correctness, quality and timeliness in hard values. They have to just provide data/values in form of theoretical validations. But the industry men are the one who deal with practical implementation of data and so they have to check each and every metric very minutely as even 1% error rate could be critical if it belongs to real life software development viz. aeronautical systems.

Software testing is an important technique for validating and checking the correctness of any kind of software. However, the production and application of effective and efficient measurement tests is not only extremely difficult, expensive and laborious but it's also a time consuming, and error prone task. As goal of software testing is to expose defects in software; as an early detection in the development cycle could not only save time and resources but along with it, an empirical evaluation of the metrics suite is also required so that user satisfaction can be achieved. In the development phase, when we talk of large-scale software development, testing accounts for a substantial larger portion of the development cost. Here, software metrics could help to

estimate the costs associated with the traditional development process. However, when reliability becomes the prime concern, then we require more efficient testing efforts for measurement and it becomes quite costly process to achieve it. So, keeping in mind above said constraints, the software testers have to develop an estimate of the total cost of the software project that includes all the work elements and procurements of the software development process.

Software cost estimation is the process of predicting the amount of effort required to build a software system [3]. Estimation of total cost could be done through determination of cost of acquisition that includes cost of training inclusive of travel and trips for customer reviews, salary of work force, and inconsistencies in the estimates. In software cost estimation process, the software requirements form the primary basis for the cost estimation. During this process, cost expenditures could be divided into three kinds of output [4]:

i. *Effort:* Amount of effort required to complete the project
ii. *Project duration:* Time needed to complete the project
iii. *Manpower loading:* Number of personnel allocated to the project as a function of time

Cost estimates will be adjusted according to available budget to arrive at final estimate.

II. COST OF SOFTWARE QUALITY

Cost of software quality includes all costs incurred in pursuit of quality or in performing quality-related activities. Cost of Quality (CoQ) includes all costs incurred in pursuit of quality or in performing quality related activities. Despite the range of definitions, the goals underlying the pursuit of quality are the same: achieving conformity, reducing variation, eliminating waste and rework, eliminating non-value-adding activity, preventing human error, increasing efficiency and effectiveness, improving productivity, and preventing defects [5].

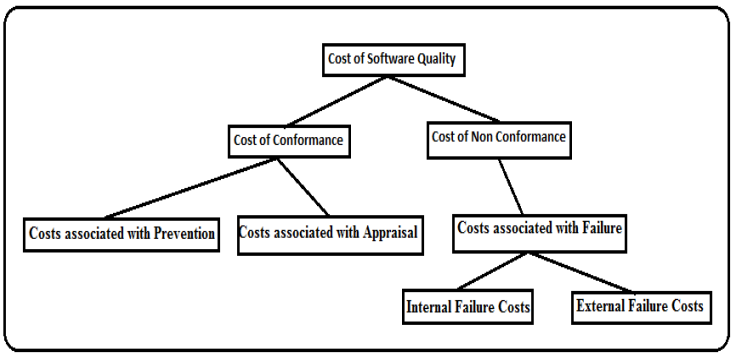

Fig. 1: Cost of Software Quality

Cost of Quality (CoQ) is useful to understand different types of costs incurred in Quality product development.

CoQ=Cost of Conformance + Cost of Non Conformance

Table 1: Cost of Quality (CoQ)

S.NO	TYPE OF COST OF QUALITY	DESCRIPTION
1	Cost of Conformance	➢ Derived from the amount developer spends on the attempts to improve quality. ➢ Conformance costs include: • *costs associated with prevention* • *costs associated with appraisal*
2	Cost of Non-Conformance	➢ Includes costs associated with failure i.e., it includes all expenses that a developer incurs when the system does not operates as specified. ➢ Non Conformance costs includes costs associated with failure i.e., • *Internal Failure Costs* • *External Failure Costs*

So, Total Cost of Quality (TCoQ) could be calculated as:

TCoQ = Prevention Cost + Appraisal Cost + Internal failure Cost + External Failure Cost

Some examples for different types of Cost of Quality (CoQ) are stated in Table 2.

Table 2: Summary of Software Cost of Quality (CoQ)

S.NO	TYPE OF COST	EXAMPLES
1	External Failure Cost	processing customer complaints, customer returns, warranty claims, product recalls,
2	Internal Failure Cost	scrap, rework, re-inspection, re-testing, material review, material downgrades.
3	Appraisal Cost	inspection, testing, process or service audits, calibration of measuring and test equipment.
4	Prevention Cost	New Product Review, Quality Panning, Supplier Surveys, Process Reviews, Quality Improvement Teams, Education and Training.

Consider the following time sheet to understand how CoQ is computed in SDLC:

Table 3: Example of Cost of Quality (CoQ) Measurement

ACTIVITY NUMBER	ACTIVITY NAME	TIME SPENT (hrs)	TYPE OF COQ
A1	Training	10-hours	Prevention Cost
A2	Requirements Review	5-hours	Appraisal Cost
A3	Requirements Rework	6-hours	Failure Cost
A4	Code Review	6-hours	Appraisal Cost
A5	Code Rework	2-hours	Failure Cost
A6	Testing	10-hours	Appraisal Cost
A7	Test Rework	5-hours	Failure Cost

From the analysis of this time-sheet data, we can easily compute the Cost of Quality as follows:

1. Prevention Cost = Cost of Activity at Activity No. (A1) = 10 hours
2. Appraisal Cost = Cost of Activities at Activity No. {(A2)+(A4)+(A6)} = 21 hour
3. Failure Cost = Cost of Activities at Activity No. {(A3)+(A5)+(A7)} = 13 hours

Therefore, Cost of Quality (CoQ) = Cost of (Prevention+Appraisal+Failure) = 44 hours

Applying the concepts of CoQ measurement, analysis and corrections consistently to the SDLC projects can help reduce the cost of quality. But, it is not the measurement, but the analysis and comparison for monitoring, control and strategic decisions that we can use the measured CoQ. To overcome the constraints, we have proposed the C.E.M: Cost Estimation Metrics in the next section.

III. PROPOSED C.E.M (COST ESTIMATION METRICS)

Software project managers are responsible for controlling project budgets; so they must be able to make estimates of how much software development is going to cost [6]. Practitioners have expressed concern over their inability to accurately estimate costs associated with software development. This concern has become even more pressing as costs associated with development continue to increase. As a result, need for a considerable research attention is in demand that is directed at gaining a better understanding of the software-development process as well as constructing and evaluating software cost estimating tools and techniques.

In real life, schedule estimation is one of the most difficult parts of the software estimation process. Three software metrics have been proposed namely; *Total Test Cost Metric (TTCM), Test and Development Cost % age Metric (TDC%M)* and *Total Software Product Cost for Testing Metric (TSPCTM).* The purpose behind proposing this metrics suite is to describe a recommended measurement and evaluation process for the development of software cost estimates by software managers.

A) Total Test Cost Metric (TTCM)

The following metric is used to measure the total cost of testing of software product and is measured by taking the average of the (CBR + CAR) i.e., sum of the total cost of testing product before and after release with LOC_{DP}, which is the total lines of code of the developed product. It is given by the equation:

$$\text{Total Test Cost metric (TTCM)} = \frac{C_{BR} + C_{AR}}{LOC_{DP}}$$

Where,
C_{BR} = Total cost of testing product before release
C_{AR} = Total cost of testing of product after release
LOC_{DP} = Total lines of code of developed product

B) Test and Development Cost % age Metric (TDC%M)

The following metric is used to measure the total test and development cost % age of software product and is measured by taking the ratio of the average of the (CBR + CAR) i.e., the sum of the total cost of testing product before and after release with (CRC + CTTM), which implies total cost of testing product from requirement to testing phases and from testing to maintenance phases respectively. It is given by the equation:

$$\text{Test and Development Cost \% age Metric (TDC\%M)} = \frac{C_{BR} + C_{AR}}{CRC + CTTM} * 100\ \%$$

Where,
CRC= Total cost of testing product from requirement to testing
CTIM =Total cost of testing product from testing to maintenance.

C) Total Software Product Cost for Testing Metric (TSPCTM)

The following metric is used to measure the total software product cost for testing and is measured by taking the average of the (CBR + CAR) i.e., sum of the total cost of testing product before and after release with (W_{WBT}+ W_{BBT} + W_{GBT}), which is the number of weighted defects found in a product under test through white-box, black-box and grey-box testing respectively. It is given by the equation:

$$\text{Total Software Product Cost for Testing Metric (TSPCTM)} = \frac{CBR + CAR}{W_{WBT} + W_{BBT} + W_{GBT}}$$

Where,
W_{WBT} = No. of Weighted defects found in a product under test through white-box testing
W_{BBT} = No. of Weighted defects found in a product under test through black-box testing
W_{GBT}= No. of Weighted defects found in a product under test through grey-box testing

As cost estimation is an important tool that can affect the planning and budgeting of a project, this projected metrics suite could help in the determination of the features that could be included within the resource constraints of the project (e.g., time). Moreover, risk of a project is reduced when the most important features are included at the beginning because the complexity of a project increases with its size, which means there is more opportunity for mistakes as development progresses. Thus, cost estimation can place a big impact not only on the life cycle and schedule for a project but also on its testing performance. It is prudent for a company to allocate better resources, such as more experienced personnel, to costly projects testing so that effective monitoring and control of the software costs is required for the verification (testing) and improvement in the accuracy of the measured estimates. So, the success of a cost estimation metrics for testing is not necessarily based on the accuracy of the initial estimates, but rather it depends on the rate at which the estimates converge to the actual cost.

IV. UNDERSTANDING THE EFFECTIVENESS AND EFFICIENCY OF C.E.M

Educating project managers, test managers, and development managers as to what we are measuring, as well as what those numbers mean is very important. This should be done for two reasons. The first is to ensure that managers support and understand the value of the metrics. It is vital that they are interested in these metrics as much as we are in providing them. The second reason is to educate them on what they can do to affect each metric positively. This last reason is the most important, yet is also the most difficult to explain. Test metrics are an important indicator of the effectiveness of a software testing process. In current years, there have been many discussions about the role of software metrics in helping software organizations to improve productivity and software quality [5]. Researchers have put much effort into learning how to use metrics for Software Process Improvement (SPI) [7] and there have been many discussions in current years about the role of software metrics in helping software organizations to improve productivity and software quality.

In this section, we'll discuss the usability of associated metrics, in order to facilitate the development of future studies as well as for measurement refinement [8]. This paper considers three significant software metrics for evaluation of cost of testing that could actually generate useful information. Software metrics are used for cost evaluation while testing, help in comprehensive evaluation, being cheaper, faster and more reliable it's easier to be used for increased efficiency, effectiveness and adoption. Proposed set of metrics can be used to predict the cost to develop & test and therefore, developers could combine the cost evaluation metric for testing to determine whether the budget allows purchasing additional computer resources that will enhance the product's quality.

The goal of the testing activity is to find as many errors as possible before the user of the software finds them. We can use testing to determine whether a program component meets its requirements. To accomplish its primary goal (finding errors) or any of its secondary purposes (meeting requirements), software testing must be applied in a systematic fashion. Testing involves operation of a system or application under controlled conditions and evaluating the results. By using our software testing metrics in a consistent manner, software developers will see improvement in the software and on the use of the metrics. However, no single metric works during all of the development phases; therefore, using several metrics for one system helps to have a handy solution that can be used during different aspects of the process of software development. Three metrics covered in this paper when used properly, i.e., when a company uses the best software testing metric during each development phase, the quality of the software will dramatically increase. Therefore, we highly recommend using software-testing metrics for the software quality assessment.

V. CONCLUSION

There are no turnkey solutions when it comes to implementing a system that will account for all of the Costs of Quality. An effective measurement activity should be able to evaluate the current process and provide suggestion to the manager for future improvement. The CEM metrics we used in our research paper could be able to provide information that is helpful for justifying the current test process. The proposed metric results clearly show the improvement that the test teams had made in the test process in terms of quality. Developing a strategy for measuring what quality costs your organization is the only way to reduce that cost, while maintaining the quality of product and retaining customers. Those companies who do it well and have the gained the competitive advantage over those that have not. Our future work includes using real industry level data to evaluate these new metrics we recommended to measure performance of individual test phases, giving suggestions to test teams and support teams for necessary changes in the test process, and implementing the whole set of metrics in a production test environment.

REFERENCES

[1]. John K. Hollmann, "Total Cost Management Framework: An Integrated Approach to Portfolio Program, and Project Management", 1st Edition, AACE® International, USA.

[2]. Latika Kharb et al., "Complexity Metrics for Component-Oriented Software Systems", ACM SIGSOFT Software Engineering Notes, Vol. 33, Issue 2 (March 2008), Article No. 4, pp. 34.

[3]. Kim Johnson, "Software Cost Estimation: Metrics and Models", 2000.

[4]. Fenton, N.E. and Pfleeger, S.L. (1997), "Software Metrics: A Rigorous and Practical Approach", International Thomson Computer Press, 1997.

[5]. Latika Kharb et al., "Reliable Software Development with Proposed Quality Oriented Software Testing Metrics", International Journal of Computer Technology and Applications, July-August 2011, Vol. 2 Issue 4.

[6]. Shaw, M. (1995), "Cost and Effort Estimation", CPSC451 Lecture Notes. The University of Calgary.

[7]. Yanping Chen, Robert L. Probert, Kyle Robeson, "Effective Test Metrics for Test Strategy Evolution", Copyright 2004, IBM Canada Ltd.

[8]. Latika Kharb et al., "AMD: Aspect-Method Dependencies Metric for Coupling", Proceedings of National Conference on Information Technology: Present Practices and Challenges, Asia Pacific Institute of IT & Management, New Delhi, August 31- September 1, 2007.