Indian Journal of Science and Technology, Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/105607, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Metrics to Develop High Quality Software

Sanjay Kumar[1*], Rahul Rishi[2] and Rajkumar[2]

[1]University Campus School MDU, Rohtak - 124001, Haryana, India; sheoranincampus@gmail.com
[2]University Institute of Engineering and Technology MDU, Rohtak - 124001, Haryana, India; rahulrishi@rediffmail.com, rajyadav76@rediffmail.com

Abstract

Objectives: The ultimate goal of the proposed metrics is to develop a quality software product, which is possible only when the product is certified at the end of each phase, so that there will be no place for errors and faults. **Methods/ Statistical Analysis**: Project A and B are "School Management Software" in C++ language. The project A is developed using the "Software quality and productivity enhancement model" procedure and verified by metrics at every phase of development and maintenance whereas project B is developed in general procedure by another group. Both the software projects are observed for six months under similar work and conditions. Metrics are applied on observations from time to time and compared. **Findings**: Results from proposed metrics certify that undefined (hidden) task in product SRS or design phase constitute the major risks associated (product failure, quality, productivity etc.) with the product. Poor requirement identification and management laid greater role in product failure. It is certified in this paper that increase in the value of product failure % decrease the quality and productivity of product. The product whose failure rate is greater than 1% is risky to use, this value varies as per the product use and its working environment. Product effectiveness curve help the user in decision making regarding the working process of the software process, if it moves downward continue the use, but if it moves upward after a time period then the product should be abort or replaced. In this paper the application of proposed metrics on small project in different phases of its development, prerelease stage and in maintenance of the product enhance the quality and productivity of the product. **Application/Improvements:** The proposed metrics are applied on small projects; they can also be applied on large, complex software products. The application of the proposed metrics reduces the risk associated with the product that enhances the product quality.

Keywords: Development Phases, Productivity, Software Development, Software Metrics, Quality

1. Introduction

Quality of software can be controlled and improved easily with the help of metrics. Metrics are that measuring units that determines the standards of any project, process and product. Metrics provides the basic comparison between the software products and also provide the information to the developer as well to the user that given software is as per the requirements or not. On the basis of reports generated by the application of metrics the software product is accepted or declined. Quality is directly proportional to reliability and reusability, as these increases the quality also increases[1]. Quality of any product affects its productivity, as quality increases productivity also increases, and this brought software metrics to the forefront[2,3]. This research paper focuses on different views on software quality. A quality product is achieved when the quality is achieved in its all phases of development. Most suitable quality model and number of metrics are applied in every phase of development starting from requirement phase then to design phase, coding phase, testing phase etc. the results at the end of each phase is calculated and evaluated, compared to increase the reliability, quality and productivity of the product. Software reusability also helps to increase the reliability, quality and productivity.

2. Object Oriented Metrics

There are number of object oriented metrics available in literature such as CK metrics, Li and Henry metrics and

*Author for correspondence

MOOD metrics[4]. The Object Oriented (OO) metrics are used to determine the quality and productivity of object oriented software products. There are a number of quality attributes that determine the quality[5,6] of software such as reusability, understability, defect density and maintainability etc. Now with the requirement of time more and more relation between these qualities attributes to be measured between them with more accuracy.

3. Motivation Behind the Proposed Work

Software quality focuses four basic areas; these are product, project quality, process and post production quality. It can be described in five different perspectives: user view, transcendental view, manufacturers view, product view and value based view[7]. The majority of the current metrics[8,9] measure the quality of product when the product is completely developed and ready to deliver. At this level of product it is difficult to make effective modification in the product, if maid then it is costly, time consuming and extra efforts and expertise. Quality must be monitored from the early phases of development such as requirement analysis, design, implementation and maintenance phases. To analyze the product size, performance, complexity, design features, performance, and the level of quality the developer uses the product metrics.

To enhance the software development and maintenance process, it is better to use process metrics. The process metrics also helps to determine the effort level and effectiveness of the development process. To determine the project cost, execution time, scheduling and productivity[10,11] of any project or product, project metrics[12] are used. There are metrics that belongs to more than one category. Daskalantonakis[13] well-articulated and developed the Motorola's software metrics program, following the work of[14], identified the goal, formulated the questions[15] in quantifiable terms and metrics were designed. A number of metrics and models are presented by software developers to enhance the quality.

4. Proposed Metrics

Project A and B are "School Management Software" in C++ language. Project A is designed as per the requirement of "University Campus School, MDU Rohtak" whereas the project B is developed by another group. The project A is developed using the "Software quality and productivity enhancement model[16]." procedure. Both the software projects are observed for six months under similar work and conditions. Metrics are applied on observations from time to time and compared[17]. To develop the quality product quality metrics are applied in every phase of development. In proposed model[16] this phase includes Define task, underline task, task defects etc. The number of undefined task (hidden task or the task that are not properly defined or understood) are studied. In the development of this project clients are involved in every phase of development. Project is certified at every phase, so that maximum errors or faults are removed at base level[18]. The hard work done in early phases provide better and easy work for next phases. The number of failure that arrives in the project due to these undefined tasks is studied and their relation is considered as initial failure rate.

Initial Failure Rate =

$$\frac{\sum_{i=1}^{n} No.\ of\ failure}{\sum_{i=1}^{n} Total\ number\ of\ undefined\ terms\ in\ the\ software\ modules}$$

$$= \frac{Number\ of\ initial\ failure\ occur\ in\ school\ management\ product}{Total\ number\ of\ undefined\ terms\ in\ school\ management\ software}$$

$$= \frac{7}{4} = 1.75$$

Objective: the objective of this metric is to find the maximum undefined terms or tasks in the product.

When it should be measured: It should be measured while designing SRS or before the design phase.

As per the requirement of the customer the SRS is designed for proper management of requirement. As the entire requirement of the customer are properly defined and as per the requirement, the risk associated with all requirement as well as to the project are studied by applying cause and effect diagram. The requirement or the processes related to the highest risk are identified and their alternate solutions are considered[16]. Before the delivery of the final SRS, fault prediction model is applied on it. It evolves the maximum faults and validates it on every boundary analysis. After completion of SRS the certification of SRS is done as

Certification of SRS Failure Rate =

$$\frac{Initial\ Failure\ Rate}{\sum_{i=1}^{n} Functional\ Specification\ of\ software\ modules}$$

$$= \frac{1.75}{41} = 0.0427$$

Objective: the objective of this metric is to find the standard of product SRS.

When it should be measured: It should also be measured in design phase or before the design phase.

The above metrics help to decide whether the SRS designed is up to the standards of the project or not. If it is ok, move to design phase otherwise review the SRS again. On the basis of the final SRS the detailed design is done. This is the phase in which logical design is converted to the physical design. A number of tools and techniques that are used to analyze the system design such as Flowchart, Data Flow Diagram (DFD), Data dictionary, Decision table and Decision tree etc. After completion of design phase, it is certified as

Certification of Architectural Design =

$$\frac{\sum_{i=1}^{n} No.\ of\ Failure\ till\ the\ design\ phase}{\sum_{j=1}^{n} Logical\ Specification\ of\ software\ modules}$$

$$= \frac{7}{93} \quad = 0.0752$$

Objective: The objective of this metric is to find the effectiveness of the design. Path testing, cause and effect diagram etc. technique can be used to find faults in design phase.

When it should be measured: It should be measured after design phase or before the design phase.

This metrics helps to validate the architectural design of the proposed project. If the design is up to the level it is moved for coding phase otherwise it is reviewed again for further necessary actions or modifications. During this phase of product development, the work is divided in modules/units and actual coding is started. This is programming phase of the project in which the programmer converts the program specifications into computer instructions. It's a crucial stage of project where the defined procedures are transformed into control specifications with the help of a computer language. Programs developed during this phase coordinate the data movements and control the entire process in project. Coding phase is certified as

Certification of coding Failure Rate =

$$\frac{Total\ Number\ of\ faults\ or\ error\ in\ all\ modules}{Total\ number\ of\ functional\ Units}$$

$$= \frac{27}{93} \quad = 0.2903$$

Objective: the objective of this metric is to find the maximum error, faults bugs in the individual modules and then in components after coding them.

When it should be measured: It should be measured in testing phase or before the testing phase.

The above metric certify the coding phase. Metrics value helps us to check the level of coding phase. As the coding phase of the product is over, it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirement phase. It brings all the pieces of the project together into a special testing environment, then checks for errors, bugs and interoperability. During this phase of development all types of functional testing like unit testing, integration testing, system testing, acceptance testing are done with non-functional testing such as defect testing, path testing etc. As the testing phase over it is certified, whether the product so obtained is up to the level or not.

Number of Failure per unit (Failure Rate)=

$$\frac{\sum_{i=1}^{n} No.of\ Bugs + \sum_{j=1}^{n} No.of\ faults + \sum_{k=1}^{n} ENo.of\ error\ j}{\sum_{p=1}^{n} Total\ Number\ of\ functional\ units\ in\ software\ modules}$$

$$= \frac{2+1+2}{93} \quad = 0.0537$$

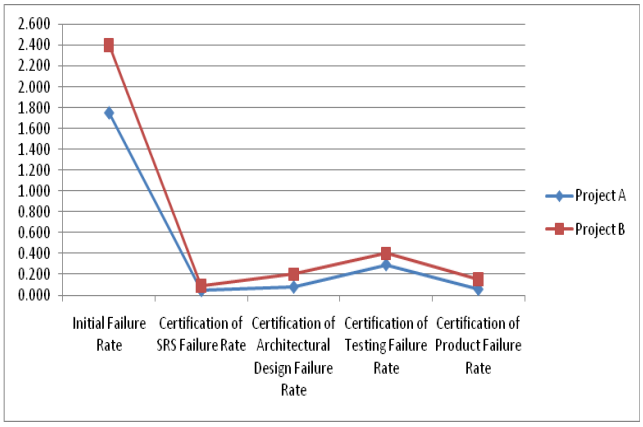

Figure 1. Comparison of school projects A and B.

Objective: The objective of this metric is to Verify and Validate the product on its boundary conditions and other inputs.

When it should be measured: It should be measured after testing phase or before the final delivery of product.

The testing team should be different from development team to find the maximum errors, bugs or faults. The above metric certify the testing phase. It verifies and

validate the project at all levels and all boundary values. When the project satisfies the entire user's requirement and is free from bugs and errors, the implementation phase begins. The Figure 1 shows the comparison of two similar projects and their certification on basis of these metrics. Project A is developed for University Campus School by using "Software Quality and productivity enhancement model" whereas Project B is developed earlier of some other school.The comparison in Figure 1 shows that the product A has lesser failure rate as compared to product B that results quality of product A is higher than that of product B.

Product Failure Rate =

$$\frac{\textit{Filure Rate}}{\textit{Total number of requirement in product}}$$

$$= \frac{0.0537}{41} = 0.0013$$

Product Failure Per cent =

$$\frac{\textit{Filure Rate}}{\textit{Total number of requirement in product}} X\ 100$$

$$= \frac{0.0537}{41} X\ 100 = 0.13$$

Objective: The objective of theses metrics is to find the standard of the final product. If the value of product failure is less than 0.01 or 1% then the product is suitable for use. This percentage also varies from product to product and its application area.

When it should be measured: it should be measured before the delivery of final product for the satisfaction of the developer and customer or user.

Implementation is that stage of a project during which theory is turned into real practice. As the product is fully developed, before the final release of the product, to enhance the quality[19,20] of the product, pre-release testing is required in the real field and real conditions. Although the developers have done their best, but still it must be assumed that something can go wrong at the worst possible moment and be prepare to switch gears quickly. Following metrics can be applied in such situations.

Fault Tracking Metrics =

$$\frac{\textit{Number of modules creating faults}}{\textit{number of modules tested to find the fault}}$$

This metric helps to track the fault and remove that fault by using minimum efforts[13].

Objective: the objective of this metric is to find the fault using minimum effort.

When it should be measured: it should be used whenever we have fault in any area and that area is not recognized.

Requirement Maintenance Metrics =

$$\frac{\textit{Number of requirement changed or modified}}{\textit{Total number of functional requirement}}$$

Objective: The objective of this metric is to maintain the maintenance record.

This metrics helps to define the maintenance work done in the product. Increased value of the requirement maintenance metric represents that the product was not up to the customer expectation.

Requirement Updation Metrics =

$$\frac{\text{(Number of requirement modified or during the month)}}{\text{(Total number of functional requirement modified + number of Functional requirement upda)}}$$

Objective: The objective of this metric is to maintain the modification and updation record.

This metrics provide the information regarding the later modification made in the product to satisfy the customer or as per required by time and conditions. .

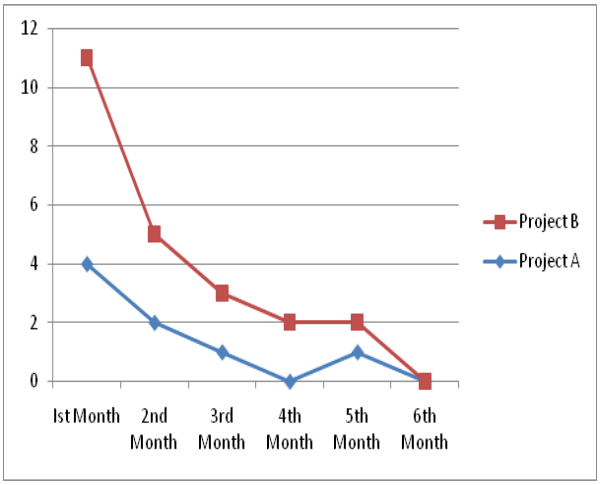

Figure 2. The number of software modification or updation in first six months of software use.

Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. The feedback is taken from the user for any modification if however required. Figure 2 shows the comparison of the projects modifications. The review of the system is done

- To know the full capabilities of product.
- To study the performance.
- To maintain the record of all updation and modifications.
- Product Effectiveness Curve:- Draw the graph of fault per month for the product ahead of implementation

Objective: the objective of this metric is to check the effectiveness of product. If the number of faults are increasing every month then

This curve helps the developer as well as user for further maintenance and for making future decision [13] regarding the product. Figure 3 show that project B has arrives more errors as compared to project A. If the curve moves downward then the product should be repaired or maintenance work should apply but if the curve moves constantly upward then the product should be replaced.

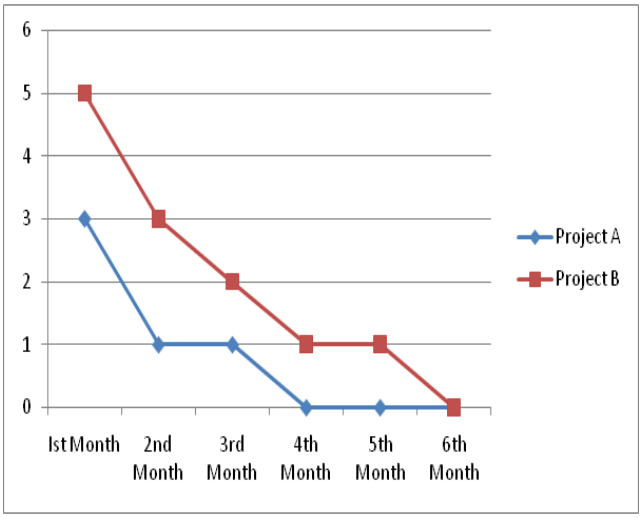

Figure 3. The number of errors in two projects.

In Figure 3, Project A exhibited lesser error or faults compared to other software.

5. Conclusion

The comprehensive results certify that project A has lesser failure rate, so project A has high quality and productivity as compared to project B. To develop a high quality product, the developer has to identify the right software development process (software development Model) as per the requirement and identify right metrics as per needs for testing and evaluation. Software with High quality being reliable, maintainable, easily integrated, well supported and portable also fulfills the needs of users and satisfies them. To develop high quality software within a limited cost and time then the developer should use good metrics from the initial phase of development that is requirement phase to avoid the maximum fault and rework. The proposed metrics applied on small project, the application of these metrics on more than ten small as well as large projects provides the numerical boundary value of each metric that helps the software developers in future to certify the characteristics of the product in every phase of the development.

6. References

1. Jethani K. Software quality- Getting right metrics, getting metrics right. Tata Consultancy Services Limited; 2008. p. 1–11.
2. Jose G, Joseph J. Test metrics and KPI's. UST Global; 2014.
3. What is more important: Software quality or productivity, Iron Triangle, Scott Ambler. Available from: http://www.ambysoft.com/essays/brokenTriangle.html
4. Kumar S. Metrics to determine the quality and productivity during software development. IJCSSE. 2016; 5(8):175–82.
5. Rajesh S, Chandrasekar A. An efficient object oriented design model: By measuring and prioritizing the design metrics of UML class diagram with preeminent quality attributes. Indian Journal of Science and Technology. 2016 Jun; 9(21). DOI: 10.17485/ijst/2016/v9i21/95147.
6. Bhatia MPS, Kumar A, Beniwal R. Ontologies for software engineering: past, present and future. Indian Journal of Science and Technology. 2016 Mar; 9(9). DOI: 10.17485/ijst/2016/v9i9/71384.
7. Stephen H, Kan K. Pearson's software quality metrics overview metrics and models in software quality engineering USA; 2002.
8. Rawat MS. Survey on impact of software metrics on software quality. IJACSA. 2012; 3(1):137.
9. Vijayalakshmi P, Luv PK, Soni AK. Rainwater runoff estimation using empirical formulae computed in c programming software for Puriliya District of West Bengal. Indian Journal of Science and Technology. 2016 Jan; 9(4). DOI: 10.17485/ijst/2016/v9i4/55229.
10. Singh G. A study of software metrics. IJCEM. 2011; 11:1–6.
11. Kaur S. Software metrics and metric tools-a review. International Journal on Recent and Innovation Trends in Computing and Communication. 2015; 3(4):2076–9.
12. Pusala R. Operational excellence through efficient software testing metrics. Infosys View Point; 2006 Aug.

13. Daskalantonakis MK. A practical view of software measurement and implementation experiences within Motorola. IEEE Transactions on Software Engineering. 1992; 18(11):998–1010.
14. Basili VR, Weiss DM. A methodology for collecting valid software engineering data. IEEE Transactions on Software Engineering. 1984; 10(1):728–38.
15. Wankhede HS, Kiwelekar AW. Qualitative assessment of software engineering examination questions with bloom's taxonomy. Indian Journal of Science and Technology. 2016 Feb; 9(6). DOI: 10.17485/ijst/2016/v9i6/85012.
16. Kumar S, et.al. Software quality and productivity enhancement model. IJERD. 2016 Nov; 12.
17. Priyadharshini V, Malathi A. Analysis of process mining model for software reliability dataset using HMM. Indian Journal of Science and Technology. 2016 Jan; 9(4). DOI: 10.17485/ijst/2016/v9i4/52931.
18. Sanjay Kumar et.al. Spiral Increment Reuse (SIR) Software Model, International Journal of Computer Science and Software Engineering (IJCSSE), Volume 5, Issue 1, 2016 January ISSN (Online): 2409-4285 www.IJCSSE.org Page: 5-10
19. Chamoli S, Tenne G, Bhatia S. Analysing software metrics for accurate dynamic defect prediction models. Indian Journal of Science and Technology. 2015 Jan; 8(S4):96–100.
20. Rashid E, Patnayak S, Bhattacherjee V. Estimation and evaluation of change in software quality at a particular stage of software development. Indian Journal of Science and Technology. 2013 Oct; 6(10).