Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 5, May 2014, pg.1303 – 1305

RESEARCH ARTICLE

Software Quality Matrics- An Overview

Sukhdev Singh Ghuman

SBDSM Khalsa College Domeli (Kpt.)

Abstract- Software measurement process is used to measures, evaluates, adjusts, and finally improves the software development process. It provides good software quality with validation and verification in software development process. There are several metrics in each of five types of software quality metrics: product quality, in-process quality, testing quality, maintenance of quality, and customer satisfaction quality.

Keywords: Software metrics, software quality, software measurement

I. INTRODUCTION

The basic purpose of quality matric is to provide quantitative information to developer during the development of software. It is also useful in some form to control the cost, schedule or quality of the software. Software metrics can be classified into three main types: product metrics, process metrics, and project metrics. Product metrics describe the characteristics of the product such as size, complexity, design features, performance, and quality level. Process metrics can be used to improve software development and maintenance. Project metrics describe the project characteristics and execution.[1]

II. QUALITY MATRICS

There are many quality matrics for assessing the quality of software. Some important ones are as discussed below:-

Defect Density Matric

The Defect Density Metric is useful for assessing the quality of requirement specifications. It looks straightforward but involves many issues. The general concept of defect rate is the number of defects over the opportunities for error (OFE) during a specific time frame. The quality of the requirements is assessed by evaluating the effectiveness of the error detection process.

Lines of Code

The lines of code (LOC) metric is anything but simple. The major problem comes from the ambiguity of the operational definition, the actual counting. In the early days of Assembler programming, in which one physical line was the same as one instruction, the LOC definition was clear. With the availability of high-level languages the one-to-one correspondence broke down. Differences between physical lines and instruction statements (or logical lines of code) and differences among languages contribute to the huge variations in counting LOCs. To calculate defect rate for the new and changed code, the following must be available:

- LOC count: The entire software product as well as the new and changed code of the release must be available.
- Defect tracking: Defects must be tracked to the release origin—the portion of the code that contains the defects and at release the portion was added, changed, or enhanced. When calculating the defect rate of the entire product, all defects are used; when calculating the defect rate for the new and changed code, only defects of the release origin of the new and changed code are included.

Function Points

Counting lines of code is a way to measure size. The second one is the function point. Both are indicators of the opportunities for error (OFE) in the defect density metrics. Now a days the function point has been gaining acceptance in application development in terms of both productivity and quality. Function points are one of the most widely used measures of software size. It is based on the functionality of the system. It takes into account what the system performs, as measure of system size. This measure is only dependent on the system capability.[2]

A function can be defined as a collection of executable statements that performs a certain task, together with declarations of the formal parameters and local variables manipulated by those statements. The ultimate measure of software productivity is the number of functions a development team can produce given a certain amount of resource, regardless of the size of the software in lines of code. A software is better if defects per unit of functions is low, even though the defects per KLOC value could be higher—when the functions were implemented by fewer lines of code. However, measuring functions is realistically very difficult.

Customer Satisfaction Metrics

Customer satisfaction is often measured by customer survey data by using the five-point scale:

- Very satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very dissatisfied

Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys.

In-Process Quality Metrics

Because our goal is to understand the programming process and to learn to engineer quality into the process, in-process quality metrics play an important role. In-process quality metrics are less formally defined than end-product metrics, and their practices vary greatly among software developers. On the one hand, in-process quality metrics simply means tracking defect arrival during formal machine testing for some organizations. On the other hand, some software organizations with well-established software metrics programs cover various parameters in each phase of the development cycle. In this section we briefly discuss several metrics that are basic to sound in-process quality management.

Metrics for Software Maintenance

When development of a software product is complete and it is released to the market, it enters the maintenance phase of its life cycle. During this phase the defect arrivals by time interval and customer problem calls (which may or may not be defects) by time interval are the de facto metrics. However, the number of defect or problem arrivals is largely determined by the development process before the maintenance phase. Not much can be done to alter the quality of the product during this phase. Therefore, these two de facto metrics, although important, do not reflect the quality of software maintenance. What can be done during the maintenance phase is to fix the defects as soon as possible and with excellent fix quality. Such actions, although still not able to improve the defect rate of the product, can improve customer satisfaction to a large extent. During the maintenance phase, the following metrics are very important:[3]

- Fix backlog and backlog management index
- Fix response time and fix responsiveness
- Percent delinquent fixes
- Fix quality

Fix backlog is a workload statement for software maintenance. To manage the backlog of open, unresolved, problems is the backlog management index (BMI).If BMI is large then 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.

The fix response time metric is usually calculated as follows for all problems as well as by severity level: Mean time of all problems from open to close. A more sensitive metrics is the percentage of delinquent fix. For each fix, if the turnaround time greatly exceeds the require response time, then it is classified as delinquent.

This metrics is not a metric for real-time delinquent management because it is for closed problem only.

Fix quality or the number of defective fixes is another important quality me

III. CONCLUSION

Software quality metrics focus on the quality aspects of the product, process, and project. They can be grouped into three categories in accordance with the software life cycle: end-product quality metrics, in-process quality metrics, and maintenance quality metrics. Product quality metrics considers Mean time to failure (MTTF) , Defect density as idea of defect rate and with metrics as Lines of count(LOC) and function points(FP). Also involvement of the customer is the important aspect of quality product. The process quality metrics considers phase-based defect arrival that is defects related to different phases of SDLC and removal pattern and defect removal efficiency (DRE) as the important quality process activity. Implementation of many such metrics of software quality help to detect errors, defects and suggest improvements for defect removal and to minimize their occurances. But none of these techniques give assurance of zero defect or defect free software.

References

[1] www.wikipaedia.com

[2] Pankaj Jalote, “An Integrated Approach to Software Engineerring” ,

[3] Ming-Chang Lee and To Chang, “Software Measurement and Software Metrics in Software Quality”