SOFTWARE MAINTENANCE METRICS 
AND ITS IMPORTANCE FOR DERIVING 
IMPROVEMENT IN SOFTWARE 
MAINTENANCE PROJECT: AN EMPIRICAL 
APPROACH

Dr. S. Ravichandran
CEO & Chief Scientist, Trimentus Technologies, 
T. Nagar, Chennai, India
Email: drravis@trimentus.com

ABSTRACT

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.

Key words: Software, maintenance and metrics

INTRODUCTION

With the advent of total quality management (TQM) organizations are using metrics to improve quality and productivity. Software maintenance organizations are no exception. In 1987 the US Navy established a centralized software support activity (SSA) to provide software maintenance for cryptologic systems. At that time two systems were supported and a software maintenance metrics program was established to support the goals of the SSA.
METRICS IDENTIFICATION
We hypothesize that the maintenance effort for a software application depends on measurable metrics that can be derived from the software development process. We first identified the metrics that could affect the effort required for maintaining an application to help managers use the right metrics. Next, we worked on establishing correlation between maintenance effort and the identified metrics.

Basic maintenance metrics
As the adage goes, you can’t improve a process without first measuring its performance, but what are the most important maintenance metrics (KPI) you should measure? Establishing a baseline for success should be the first step whenever you set out to improve something. Bill Gates draws the example of the steam engine – one of the biggest advancements in the industrial age, as being a product of incremental design changes and precise feedback, as opposed to one Eureka! – type moment. “Without measurement,” writes William Rosen, invention is “doomed to be rare and erratic.” With the steam engine, the criteria is fairly straightforward. A superior design would have some combination of being lighter, more powerful, more fuel efficient, cheaper to construct, etc.

The same is true for maintenance metrics and there is a wealth of performance indicators that can be used to measure and improve performance. For example,

- % Preventive/Reactive maintenance
- % of work orders completed on time
- Downtime hours/month
- $ of Spare Parts consumed

Minimizing downtime might seem like a worthy goal, but not if it also has a negative impact of product quality or employee morale. Or, if it jeopardizes a longer term strategy of increasing the proportion of preventive/reactive maintenance. When optimizing the department there are dozens of confounding factors to be considered. The struggle then becomes which maintenance metrics to focus on.

SOFTWARE MAINTENANCE METRICS
This section describes the set of metrics that were selected because they are among the most used and well known.

Maintainability Index
The Maintainability Index (MI) has been proposed for assessing the maintainability of complete systems. The original three-metric MI uses a polynomial to combine the average per module of three traditional code measures (lines of code, cyclomatic complexity and Halstead Volume) into a single-value indicator of maintainability. An improved four-metrics version of MI also includes the number of comments.
Software Maintenance Metrics and Its Importance For Deriving Improvement In Software Maintenance Project: an Empirical Approach

**Structural Measures**
The most common set of metrics for assessing code maintainability is structural measures (SM), including the CK metrics. The subset includes the coupling measure OMMC (call to methods in an unrelated class), the cohesion measure TCC (tight class cohesion) and the measure of size of classes WMC1 (number of methods per class; each method has a weight of 1) and depth of inheritance tree (DIT).

**Code Smells**
The concept of code smells was introduced as an indicator of problems with the software design. Code smells have become an established way of indicating issues with software designs that may cause problems for future development and maintenance. However, a systematic review [18] found only five studies that investigated the impact of code smells on maintenance.

**Classification of Software Metrics in Software Engineering**
Measurement is done by metrics. Three parameters are measured: process measurement through process metrics, product measurement through product metrics, and project measurement through project metrics.

![Classifications of software quality metrics](image_url)

**Process Metrics**
To improve any process, it is necessary to measure its specified attributes, develop a set of meaningful metrics based on these attributes, and then use these metrics to obtain indicators in order to derive a strategy for process improvement.

**Product Metrics**
In software development process, a working product is developed at the end of each successful phase. Each product can be measured at any stage of its development. Metrics are developed for these products so that they can indicate whether a product is developed according to the user requirements. If a product does not meet user requirements, then the necessary actions are taken in the respective phase.
Project Metrics

Project metrics enable the project managers to assess current projects, track potential risks, identify problem areas, adjust workflow, and evaluate the project team's ability to control the quality of work products. Note that project metrics are used for tactical purposes rather than strategic purposes used by the process metrics.

MAINTAINABILITY METRICS

The model proposed here, which has been used in case studies, considers only three software characteristics. Each one directly affects one maintainability component. XU: understandability metric The number of comment lines for every 100 lines of code. We observe that there is a close relationship between the internal documentation of the code and understanding cost. As expected, as the value of the understandability metric increases (number of comment lines), the understandability index (directly proportional to understandability cost) decreases. XM: modifiability metric The number of lines without constant data for every 100 lines of code. We observe that the more numerous the constant data in the code, the bigger the modification cost. XT: testability metric The number of error testing lines for every 100 lines of code. We observe that testing the code will be simpler if there are error detection and treatment procedures built into the code. These three characteristics have been chosen because we observe that they are easily measured, and they have a great influence on maintainability. Nevertheless, the model can be applied whatever the metrics chosen, provided the interdependence between each metric and its maintainability component can be demonstrated.

USE OF METRICS TO IMPROVE SOFTWARE MAINTAINABILITY

Software maintainability is a difficult factor to quantify. However, it can be measured indirectly by considering measures of design structure and software metrics. It is also claimed that logical complexity and program structure have a strong correlation to the maintainability of the resultant software. Moreover, as Fenton says “Good internal structure provides good external quality”. But what must be determined is up to what point and in which cases can we rely on software metrics in order to define the maintainability of a software product.

A software company that wants to apply a framework of software metrics in order to control the level of the maintainability of its products must firstly choose a set of appropriate metrics for each product. Secondly, it must arrange the boundaries of each metric that characterise a code as accepted, as required for corrections or as rejected. Then, after conducting the internal measurements, if any module or part of the program fails according to the predetermined metrics limits, it must be inspected by the programmers in order to decide the appropriate step. These emergent interventions prevent the programming team from facing serious problems and making great efforts during the maintenance phase. Moreover, if the metrics’ results arise great deviation from the accepted level in the majority of the modules, the software product is preferable to be re–designed. Therefore, project managers must be particularly mindful when they determine the constrains of each metric used in the applied framework.
CONCLUSION
Software metrics provide an easy and inexpensive way to detect and correct possible causes for low product quality according to the maintainability factor as this will be perceived by the programmers. Setting up measurement programs and metric standards will help in preventing failures before the maintenance process and reduce the requisite effort during that phase. Internal metrics are highly correlated with the programmers’ opinion of maintainability. However, dissatisfaction with internal quality standards may not necessarily result in low level of maintainability although it is usually expected. In that case, it is preferable that, despite what internal measurements indicate, the final judge for the maintainability of the produced software is the programmer.

REFERENCES