IOSR Journal of Computer Engineering (IOSR-JCE)
ISSN : 2278-0661, ISBN : 2278-8727, PP : 50-55
www.iosrjournals.org

Software Engineering Metrics: Introduction

Rahul N. Lokhande[1]
[1]*(D.Y.Patil College of Engineering and technology, Kasaba Bawada, Kolhapur.)*

ABSTRACT*:* *Constructing validity in the software systems are moving around the questions, What efforts have to take for analyzing the software system and to evaluate the performance of the system? The discussion for constructing validity starts with the discussion of measuring the attributes of the software system which are known as the software metrics. This paper presents the introduction to the software metrics as the key point for evaluation of validity, performance , analysis and protection from faults in the software systems. There is also classification of the software metrics, collection of metrics using software metric tools. A model is proposed for fault detection using software metrics, this paper also enlighten various applications of software metrics*.
Keywords– *Fault detection , lines of code,OOO meter ,Software metrics, Software metrics tools*

1. INTRODUCTION

Software measurements previously are limited to measuring individual, product software attributes. Rather than this one dimensional approach towards measurements, organizations are coming forward with integrating complete software metric programs into the software development processes. Measuring the software metrics is not only for process improvements but also to reach higher Capability Maturity Model levels.
Software metrics are the attributes of the software systems that deals with the measurements of the software product and process by which it is developed.
Properties of the metrics:

o *simple, definable*: The metrics should be simple and definable so we get a clear idea about how the metrics can be evaluated.
o Objective: The metrics should be objective to the greatest extent possible
o Easily available: The metrics should be easily available that is the cost to get the metrics should be reasonable (i.e. related to cost).
o Valid: The metric should measure what intended to measure.
o Robust: The metric should be relatively insensitive to insignificant changes in the process or product.

The effective management of any process requires quantification, measurements and modeling. Software metrics provide a quantitative basis for the development and validation of the software development process. Metrics can be used to improve software productivity and quality. The metric measurement models introduce the most commonly used software metrics and reviews their use in constructing models of software development process and can be handled by improving software management capabilities. Many software metrics that are invented and they have defined and tested in limited environments. The search for a commonly accepted set of software metrics to be measured is still on.
The paper is organized as follows: section II describes the classification of software metrics, section III represents the tools to collect software metrics, section IV focuses on applications of software metrics and how the collection of these metrics will be beneficial, section V proposes a model of fault detection using software metrics and section VI draws the conclusion.

II. HEADING S

1. Introduction
2. Classification of Software Metrics
3. Software metric Tools
3.1 Analyst4
3.2 CCCC
3.3 Chidamber & Kemerer Java metrics
3.4 Dependency Finder
3.5 Eclipse Metrics plug in1.3.6.

3.6 Eclipse Metrics plug in 3.4
3.7 OOMeter
3.8 Semmle
3.9 Understand for Java
3.10 VizzAnalyzer
4. Application of Software metrics
5. Fault detection using software metrics
6. Conclusion

III. INDENTATIONS AND EQUATIONS

3.1. Introduction

Software measurements previously are limited to measuring individual, product software attributes. Rather than this one dimensional approach towards measurements, organizations are coming forward with integrating complete software metric programs into the software development processes. Measuring the software metrics is not only for process improvements but also to reach higher Capability Maturity Model levels.

Software metrics are the attributes of the software systems that deals with the measurements of the software product and process by which it is developed.

Properties of the metrics:
o *simple, definable*: The metrics should be simple and definable so we get a clear idea about how the metrics can be evaluated.
o Objective: The metrics should be objective to the greatest extent possible

o Easily available: The metrics should be easily available that is the cost to get the metrics should be reasonable (i.e. related to cost).
o Valid: The metric should measure what intended to measure.
o Robust: The metric should be relatively insensitive to insignificant changes in the process or product.

The effective management of any process requires quantification, measurements and modeling. Software metrics provide a quantitative basis for the development and validation of the software development process. Metrics can be used to improve software productivity and quality. The metric measurement models introduce the most commonly used software metrics and reviews their use in constructing models of software development process and can be handled by improving software management capabilities.

Many software metrics that are invented and they have defined and tested in limited environments. The search for a commonly accepted set of software metrics to be measured is still on.

3.2. Classification of software metrics

Software metrics are broadly classified as
1. Product metrics
2. Process metrics

Product metrics are measures of the software product at any state of the product's development, from requirements to installed systems. These metrics measures the complexity of the software design size of the final program (source or object code), number of pages of documentation produced.

Process metrics are the measures of the software development process. The examples of the process metrics are as methodology used, average level or experience of the programming staff.

Another classification of software metrics is as follows:
1. Objective metrics
2. Subjective metrics

Objective metrics always results in identical values for a given metric as measured by two or more qualified observers. Whereas subjective metrics are those that even qualified observers may measure different values for a given metric since their subjective judgment is involved in arriving at the measured value.

For product metrics the size of the product measured in lines of code (LOC) is an objective measure. The example of the subjective etrics is classification of software as “organic”,“semi-detached” or “embedded” as required in the COCOMO cost estimation model. For the process metrics, development time is an example of an objective measure and a level programmer experience is a subjective measure.

Third way to classify software metrics is primitive metrics and computed metrics. Primitive metrics are those that can be directly observed such as lines of code (LOC) , number of defects in unit testing or total development time for the project. Computed metrics are those that can not be directly observed but are computed in some manner from other metrics. Examples of computed metrics commonly used for productivity is lines of code (LOC) produced per person-month (LOC/ preson-month) and those used for product quality is number of defects per thousand lines of code (defects/KLOC).

Computed metrics are the combination of other metric values and thus are often more valuable in understanding or evaluating the software process than the simple primitive metrics. Although software metrics can be nearly categorized as per the classifications mentioned above, the models does not follow these organizations.

3. Software Metric Tools

This section briefly describes the software metric tools for the collection of metrics. There are number of software metrics have been developed and numerous tools exist to collect the metrics from program representations. This large variety of tools allows a user to select the tool best suited as per the use requirements for example it's handling,tool support ,cost etc. This is assumed that the metrics computed by the metric tools are same for all the metric tools.

One can think of a software metric tool as a program which implements a set of software metrics definitions. It allows to access a software system according to the metrics by extracting the required entities from the software ad providing the corresponding metric values.

There are some criteria for selecting the proper metric tools as the availability of the software tools can make confusion. One such criterion is that the tools must have to calculate any form of software metrics. Majority metric tools are available for Java programs. Many tools are just code counting tool, they basically count the variants of the lines of code(LOC) metric. The specific criteria are as follows language: Java(source or byte code), metrics: well known object oriented metrics on class level, license: freely available. Following are some software metric tools with the criteria mentioned above:

3.1 Analyst4

It is based on the Eclipse platform and available as a standalone Rich Client Applications or as an Eclipse IDE plug in. It features search, metrics,analyzing quality,report generating for Java programming.

3.2 CCCC

It is an open source command line tool. This tool analyzes C++ and Java. It generates reports on various metrics including lines of code (LOC) and metrics proposed by Chidamber & Kemerer and Henry & Kafura.

3.3 Chidamber & Kemerer Java metrics

It is open source command line tool.It calculate the C & K object oriented metrics by processing the byte code of the compiled Java files.

3.4 Dependency Finder

It is a open source command line tool. It is a suite of tools for analyzing compiled Java code. It's core function is a dependency analysis application that extracts dependency graphs and collect them for useful information. This tool also comes in forms such as a swing based application a web application and set of ant task.

3.5 Eclipse Metrics plug in1.3.6.

It is an open source metrics calculation and dependency analyzer plug in for the Eclipse IDE. It also measures various metrics It detects cycles in package and type dependencies.

3.6 Eclipse Metrics plug in 3.4

It is a open source tool. It calculates various metrics during build cycles and warns via the Problem view of metrics range violations.

3.7 OOMeter

It is an experimental software metrics tool developed by Alghamdi et. al. It accepts Java/C# source code and UML models in XML and calculates various metrics.

3.8 Semmle

It is an Eclipse plug in. It provides an SQL like querying language for object-oriented code which allows to search for bugs, measure code metrics etc.

3.9 Understand for Java

It is a reverse engineering, code exploration and metric tool for Java source code.

3.10 VizzAnalyzer
It is a quality analysis tool. It reads software code and other design specifications as well as documentations. It also performs number of Quality Analysis.

4. Applications of Software Metrics

In this section, I talk about the various applications for which the collection of software metrics is necessary. Measuring software metrics facilitate private self assessment and improvement. Metrics in hands will help to evaluate project status, staff performance. Metrics also informs about the characteristics of the product. Metrics will be used in counting purposes such as counting bugs,reported reported hours,branches, lines of code .
Metrics used for matching difficulty or complexity of product can be estimated by matching it to one of several products already completed. Comparing is the purpose of metric collection where on can say one specification item is more clearly written than another. Timing, for getting the various times in the software system is possible with help of metrics such as measured the time until specified events, time required to complete the task.
A metric might be expressed as a formula involving more than one variable, such as Defect Removal Efficiency(DRE) which is computed as the ratio of defects found during development to total effects.A more useful and important application of the metrics collection is for Fault detection which is described in next section.

5. Metrics for fault detection

There is a co-relation between the object oriented metrics and the faults found in the classes, that is with the regular measurement and analysis of object oriented metrics, the qulity and the maintainability of a software system can be improve and its testing can be made more efficient
A major challenge for the software management is to detect and isolate the software faults effectively from software systems. A large amount of metrics can be collected from the system components for fault analysis. metrics collected can be used for the purpose of system monitoring and fault detection.
Previous work for the fault detection using software metrics are based on the metric co-relations are variants based on the Jaccard coefficients. These techniques are evaluated related to the metric-pair methods, which are so tedious to find the metric pair.
In this paper I am proposing a model for fault detection using software metrics. The basic idea of the model goes like this. First metrics will be collected using the software metric tools described earlier in Section III or one can write his/her own code to collect the metrics for more flexibility in collection. The metrics will b collected with some rules such as, the collected metrics should not have any variants, they should be collected from the software systems at fault free time.
Computing Similarities: This module consists of computing the Empirical Entropy and empirical Conditional entropy to compute the Normalized Mutual Information (NMI), which is a standard similarity measure for metrics. The Similarity matrix of the metrics is created using NMI.
The Empirical entropy H(X) for metric X is computed using the formula:

$$H(Y|X) = -\sum_{i}\sum_{j} \frac{n_{ij}}{n}\log\frac{n_{ij}}{n_i}$$

……………..(1)

Where, n: number of samples observed

k: number of bins in which samples divided

n_i: number of samples observed in bin i

The Empirical conditional entropy H (Y|X) is computed using the formula:

$$H(Y|X) = -\sum_{i}\sum_{j} \frac{n_{ij}}{n}\log\frac{n_{ij}}{n_i}$$

Where $n_{ij:}$ number of samples (x, y) with x in bin i and y in bin j.
The Normalized Mutual Information (NMI) is computed using the formula:

$$NMI(X,Y) = \frac{H(Y) - H(Y|X)}{\sqrt{H(X)H(Y)}}$$
……………….(3)

Clustering: This module is for the purpose of clustering the metrics in standard manner. By getting the similarity matrix from the above step, the Complete link hierarchical agglomerative clustering is applied to group the similar metrics. This algorithm works by grouping the metric one by one on the basis of nearest distance measures of all the pair wise distance between the metrics.
Algorithm for clustering:
Steps:
1. Take similarity matrix M as a input
2. Define the threshold for maximum distance between two clusters.
3. Define the maximum distance between the two clusters as the distance between the two clusters.
4. Treat every metric as a single cluster and merge nearest cluster until either the distances between every two clusters exceed the predefined threshold or all metrics belong to one cluster.
The algorithm gives the guarantee about all metrics in cluster have similarity of the predefined threshold. The algorithm gives facility that the specification of clusters a priori is not necessary. *Computing In-cluster Entropy:* This module computes the status of the each cluster in the form of entropy. The metrics in one cluster are closely correlated with each other since, the relationships between the metrics are either linear or nonlinear it is difficult to establish analytical models for the cluster or for the metrics in the cluster.
The empirical entropy can be taken as the signature of the cluster, which provides the status of the cluster. For a given cluster, a significant change in the behavior of the cluster entropy indicates a fault.
The steps for computing cluster entropy:
1. Normalization of all metric values by dividing each value with its average values which is based on data collected during normal operations.
2. Relate the different metric values to a single variable.
3. Calculate the empirical entropy of the random variable over time.
1. Set the range to[0,7]
2.Divide it into seven equal bins.
3.Add an eighth bin with the range [7, ∞], for the values that do not fit in the other seven bins.
4.Monitor the cluster entropy over time i.e. tracking the entropy of each cluster.
Fault Detection : In this module nonparametric statistical test namely Wilcoxon Rank-Sum Test is used to identify significant shifts in In-cluster entropy of individual clusters for detecting the fault within the systems.
Let Ei be the In-cluster entropy of cluster E at time i. For detecting the significant change in Ei when fault occurs, two sliding windows of Ei are kept. The test window consists of most recent nEi's and the baseline window consists of mEi's before the test window. Then applying the
Wilcoxon Rank-Sum Test on these windows. If the significant change is indicated by the test alarm is raised.
For example, if $\{E_{s+1}, E_{s+2},\dots,E_{s+m}\}$ and $\{E_{s+m+1}, E_{s+m+2},\dots,E_{s+m+n}\}$ are two sample sets from two sample windows (s+1,s+m) and (s+m+1,s+m+n) are from the same distribution, then
Wilcoxon Rank-Sum statistic is given by the formula:

$$W = \sum_{i=1}^{m}\sum_{j=1}^{n} h_{s+i,s+m+j} + \frac{m(m+1)}{2}$$
………………..(4)

Where

$$h_{i,j} = \begin{cases} 1, & X_i < X_j \\ 0.5, & X_i = X_j, \\ 0, & \text{otherwise.} \end{cases}$$

IV CONCLUSION

As the software engineers, developers must be able to rely on the results, especially the results on the software quality engineering and metrics should be reliable. Software metrics are used during forward engineering approaches it is necessary to take early measures if some parts of a system deviates from the specifications given earlier, or during maintenance activates and to identify parts of the software system needing attention.
Some of the typical metrics are discussed and reasoned about for years , but only few of them are actually taken under consideration for experimental work. Software engineers should be able to rely on the tools implementing these metrics to support them in quality assessment and assurance tasks to allow to quantify software quality and to deliver the information needed as input for their decision making and engineering processes. Nowadays a large amount of software metric tools are available, but it would be necessary that all metric tools implement the suggested set of metrics the way they have been validated.

REFERENCES

[1] Cem Kaner, "Software Engineering Metrics : What do They Measure and How Do We Know?", 10th International software metrics symposium, Metrics 2004.
[2]Rudiger lincke,Jonas Lundber and Welf Lowe, " Comparing Software Metrics Tools".
[3] Everald E. Mills, "Software Metrics",SEI curriculum Module SEI-CM-12-1.1, Dec 1988.
[4]Alberto Sillitti, Barbara Russo, Paolo Zuliani, Giancarlo Succi, "Deploying, Updating, and Managing Tools for Collecting Software Metrics".
[5]Tracy Hall, Norman Felton, " Implementing Effective Software Metrics Programs" IEEE Software,1997.
[6] Istevan Siket, "Applying Software Product Metrics in Software Maintenance",2010.
[7] Z.Guo, G. Jiang and K.Yoshihira, "Tracking Probabilistic Correlation of Monitoring Data for Fault Detection in Complex Systems", *Proc int'l Conf. Dependable Systems and Networks (DSN '06). 2006*.