Abstract – A number of individual practitioners and organization adopt Open Source Software (OSS) through try and error approach which leads to the problems of coming across software and then abandoned after realizing lack of important qualities to fit their requirements or facing difficulty in maintaining the software. Several software metrics for measuring object oriented software development process have been proposed but contribution to evaluate readymade OSS is still limited. This paper provides an overview of current available metrics and their applicability in OSS environment.

Keywords: open source software; Object oriented software; maintainability.

I. Introduction

Over the past three decades, the growth of Free Open Source Software (OSS) has triggered individuals, public and private organizations to adapt it in plenty. For that reason, OSS developers have consistently being releasing much software in the market [1] as an attempt to win available opportunities characterized by the presence of many firms that have shifted to economic strategies from commercial software adoption.

With so many Open Source Software (OSS) products in the Market, organizational managements struggle for reliable information that suit their software needs. However, [2],[3] argued that the software market situation is obliges OSS to have reasonable quality that authenticates their increasing adoption. Many organizations face decision making challenges on adopting this technology because there is little objective evidence to confirm the quality and inherent risk. Thus, flourishing, compliance, user subjective evaluation and successful stories of the benefits sought to influence decisions of the most practitioners in the Open Source discipline [4], [5], [6].

This leads to strong maintainability challenges that force individual practitioners and organizations to abandon software after coming across the one that fit more compared to the previous or realizing the implemented software missing some maintainability quality attributes. As it is widely acknowledged that the maintenance is complicated and expensive parts to handle in software development rotation. In the software development life cycle, maintainability is assumed to consume more than 40 to 70 percent of time and resources [7],[8] entails by the effort of utilizing software for a short period of time.

To solve the stated crisis, the metrics for predicting maintainability of OSS is required in ensuring desired quality of software. Moreover, the quality demand in OSS technology provide the consent for researchers to work more in this body of knowledge regardless of a limited role of available metrics in OSS environment due to the nature of its development philosophy.

In this review paper, the researchers intend to present the early stage of an ongoing research to propose the metrics model that could be used by OSS practitioners in predicting maintainability. This can build their confidence in making the right decision in selecting OSS for their needs.

Specifically, this paper is organized as follows; Section II reviews Object Oriented Metrics. Section III compares different Object Oriented metrics and examines those that can be used to measure Open Source Software. Section IV shows the analysis of Open Source software metrics that can predict maintainability. Finally, Section V concludes the study.

II. Object oriented metrics

As in a number of disciplines, metrics is an important indicator to provide information for the people to make more informed decisions and intelligent choices. To meet up the accomplished quality in software development, the metrics should be employed in every stage from the requirement collection, analysis, design, the implementation and maintenance in order to assist the developer in making decision for certain performance. Fenton and Pfleeger [9] divide software engineering rotation metrics in terms of process, products and resources.

In determining the relation between different traditional and object oriented metrics and how they
can be used as an indicators of complexity of the software, different object oriented metrics has been studied. Evidence shows that some oldest metrics used by traditional software development were also suited to measure a newly development approach. The primitive metrics like Line of Code (LOC) were used in both technology phases. It can be directly observed through counting the individual lines of code per product [9]. However, this metrics face criticism on whether blank and comment lines should be included or not, but still some researches recognize this as the legitimate method of measuring the size of software products[10], [11]. Therefore maintainability differences are observed between software with more code and that with fewer codes. Thus, LOC metrics is still useful in Object oriented as well as traditional approach.

Other object oriented metrics for measuring quality of software products was also explained by [12] who mentioned three metrics suits in predicting the quality of packages in OOS. The first group contains six related metrics of Number of Children (NOC), Depth of Inheritance Tree (DOTT), Lack of Cohesion in Method (LCOM), Response for a Class (RFC), Coupling Between Object (CBO) and Weight Method per Class (WMC). This sets of metrics measure size, coupling and inheritance using packages which is a key component in OOS [11], [12], [13], [14]. The second collection explains five metrics that work on size, coupling and stability of different attributes of software packages [11]. Authors conducted an experiment of analyzing number of classes included inside the package and introduced metrics based on package. These metrics are Afferent and Efferent coupling, packaging metric in elastic to change and Distance metrics that measure package balance between abstraction and stability.

The last cluster called Metrics for Object Oriented Design (MOOD) set. It was assigned to measure the trends of OSS design phase [10][15], these include Method Hiding Factor (MHF) and Active Hiding Factor (AHF) working in encapsulation property, Method Inheritance Factor (MIF) and Attribute Inheritance Factor (AIF) based on inheritance and finally Polymorphism Factors (PF) and Coupling Factor (CF) that is based on message-passing between objects [8][12]. These sets of metrics seemed important in captures nearly all basic structural properties of Object oriented paradigm.

Exceptional handling is the latest metrics to be studied in measure immunity of software codes in object oriented software development. In their study, [16], introduce a Number of Catch blocks per Class (NCBC) and Exception Handling Factor (EHF) as two metrics to measure robustness of OOS in the block of exceptional handling. In validating these metrics, authors use Weyuk’s structural programming technique to confirm their metrics [17], [18]. These two metrics can be used by software evaluation team to check how errors have been handled before software being used. This property is remarkable in ensuring OSS adaptation in accomplishing maintainability quality attribute.

Another important code quality metric is Cyclomatic complexity which judges how hard to maintain a given code and how likely the code may produce errors [19]. In this case, complexity of product is determined by the amount of decision made in the source code [15]. In another novel, [21] in trying to identify quality of object oriented projects codes from group of metrics, and out of them they acknowledged Length of Class Names (LCN) and Lack of Documentation (LD) as metrics to measure OO projects, this means that project with more documentation seemed to be more quality and easy to maintain, although there was no enough evidence to prove relationship of LCN and the quality of software.

The Table 1 below listed some of the most popular metrics to measure software quality.

### Table 1

**Table I**

**Popular Products Metrics that can be used to measure software quality**

<table>
<thead>
<tr>
<th>Metrics</th>
<th>OO Metric</th>
<th>Applicable in OSS Metrics</th>
<th>Applicable in measuring OSS Maintainability</th>
<th>Article being presented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coupling Between Object (CBO)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[12][5][25]</td>
</tr>
<tr>
<td>Depth of Inheritance Tree (DIT)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[12][5][22][24]</td>
</tr>
<tr>
<td>Lack of Cohesion In Method (LCOM)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[12][5][24][34][36]</td>
</tr>
<tr>
<td>Number of Children (NOC)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[12][5][24]</td>
</tr>
<tr>
<td>Response for a Class (RFC)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[12][5][24]</td>
</tr>
<tr>
<td>Weighted Method Count (WMC)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[12][5][24][26]</td>
</tr>
<tr>
<td>Data Abstraction Coupling (DAC)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[25][24]</td>
</tr>
<tr>
<td>Message Passing Coupling (MPC)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[11][24]</td>
</tr>
<tr>
<td>Number of Local Methods (NOM)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[25]</td>
</tr>
<tr>
<td>Number of Attributes and Methods (NAM)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[24][25]</td>
</tr>
<tr>
<td>Tight Class Cohesion (TCC)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[26]</td>
</tr>
<tr>
<td>Locality of Data (LD)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[23]</td>
</tr>
<tr>
<td>Improvement of LCOM (LCOM)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[23]</td>
</tr>
<tr>
<td>Lines of Code (LOC)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[19][24][31]</td>
</tr>
<tr>
<td>Lack of Documentation (LD)</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>[23][24][25]</td>
</tr>
<tr>
<td>Length of Class Names (LCN)</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>[26][23][24]</td>
</tr>
</tbody>
</table>

Halstead’s Metrics (Program Vocabulary,
III. Open Source Software Metrics Overview

This section will provide an overview of OSS metrics that can be used to measure the maintainability attributes of OSS. Open Source Software development based on early released of the product, therefore practitioner who intended to employ these kind of software has extra effort to stabilize the product. The existing evidence that maintainability can consume forty to eighty percent of development effort gives special attention in dealing with in this body of knowledge. In accomplished this study, the earlier results has shown that many metrics that are working in object oriented technology seemed to fit in open source software technology. Lack of information about early development phases in OSS and the wider availability of source codes force the OSS practitioners to assess the quality of software based on attributes related to the codes. Moreover, professionals in this discipline remain with the only choice of working directly to the metrics associated with products after release. Metrics such as size, inheritance, coupling and Information flow metric proposed [21] proved to have properties that may work directly to the product of which could be used as indicators in various OSS products [28][29][31][19][34]. Also the uniqueness of OSS expelled some of the metrics that used in object oriented technology to apply to OSS. Examples Lack of Documentation (LOD) and Component flow metrics are not fit to be used to measure OSS. This is the fact that OSS is not provided with documentation and the complication evidence of lack of documentation was observed in [33] when authors conducted statistical validation of some metrics in OSS. Investigators faced challenges of conducting manual search to identify the specific class that used wild card library such as import java.io.* and component flow metrics contradict with the OSS code based philosophy. This metrics work on component environment where most of them, code accessibility is difficult [29], so this is then another outlier metrics within the OSS discipline.

IV. Open Source Software Metrics that Predict maintainability

In traditional software development, maintenance considered during early phase of software development process, developers take this as crucial part in order to reduce cost and effort of maintaining the system [34][33], therefore at this phase, the closeness between the vendor and the client are highly emphasized. However, this prominence is important in capturing correct requirements from the client in order to develop quality product that can reduce maintenance effort. Unfortunately, the practice of Open Source Software is different due to the absence of direct connection between the developers and the clients. Practically, developers from unknown location contribute to the online software without management profile [36]. Thus the client does not know whether the developers have considered important aspects of software development philosophy like maintainability during design phase. The study has identified practical evidence showing various object oriented metrics could possibly be used to predict maintainability in this new key technology. Metrics for measuring properties like inheritance, size, complexity, cohesion and coupling proved to determine the effort of maintainability [10], [13], [14], [23]. For example inheritance promotes software reuse but in one side can create the possibility of violating important properties like encapsulation and information hiding since subclass need to have some access to super class. Moreover, changes made during maintaining phase can sometimes increases the depths of inheritance.
tree by adding more offspring. Therefore, by assessing the inheritance tree available in the product, it is easy to predict how much effort needed to make it stable.

Measuring software size using LOC has been acknowledged for long time in software development discipline [13]. Software with more line of code is more complex compare to that with few lines, thus, LOC metric is also useful in OSS as the code is the only resource available by practitioner to make changes while he left without any documentation.

Coupling between object is also another cause of complexity in maintaining the software [37], [11]. Changes done in one object should also affect the partner object. This is due to the dependency between the object of the class. Therefore, product that has implement more coupled object is difficult to maintain.

Cohesion is an interchangeable property of coupling. It is the extent on which members of the class related and shows interdependency within modules boundary. Thus, increase reusability of the member within a class, promote changes and experience undesirable side effect when making those changes [25]. Cohesion favors the maintainability of software, where high cohesive class is more understandable, maintainable and easy to modify.

IV.1. Maintainability metrics in OSS

The properties applied in software design can result in complicating the product. Complexity determines the effort required by practitioner in handling the software [19], [38]. Figure 1 shows the relationship between metrics, complexity and maintainability of software.

Upon combining above complexity comparison, equation 2

\[
\text{Maintainability effort} = \frac{1}{(\text{LOC} \times \text{NAM} \times \text{DIT} \times \text{CBO} \times \text{RFC} \times \text{LCOM} \times \text{LOC} \times \text{CC} \times \text{EHF} \times \text{NCBC})}
\]

Where constant, k is OSS context being implemented.

V. Conclusion

This article reviewed Object Oriented Software metrics from different literatures in order to come up with model that can be used to predict maintainability of OSS. Study shows that many object oriented software metrics can also be used to measure properties in OSS but some few metrics like LOD and LCN are not applicable in predicting maintainability of this approach. This is ongoing
research aim on proposing OSS model to predict maintainability. This paper presents the relationship between available metrics and its applicability to OSS maintenance, future work will identify metric that will suitable for particular OSS.

References


**Authors’ information**

**Abubakar D. Bakar** Born in Zanzibar, Tanzania, in 1977. He obtained his B.Sc. in Computer science in 2004. He received his M.Sc. in Computer Science at University of Dar es salaam Tanzania, 2010. Currently, he is PhD student in the Faculty of Computer Science and Information System at University Putra Malaysia (UPM). His research interest in Search based Software Engineering. Mr. Bakar is a member of software engineering group in the Faculty of Computer Science at University Putra Malaysia.

**Abu Bakar Md. Sultan** was born in Melaka, Malaysia in 1965. He holds a PhD in Artificial Intelligence from University Putra Malaysia (UPM) in 2007. His research focus is in Artificial Intelligence and Software Engineering particularly Search-based Software Engineering (SBSE). He has published articles in conferences and various journals related to SBSE. Associate Professor Dr Abu Bakar Md. Sultan currently is the head of Information System Department, Faculty of Computer Science and Information Technology, UPM.

**Zulzalil H.** was born in Kuching, Sarawak on the 28th of February 1972. She received her bachelor degree in computer science from University Agriculture of Malaysia in 1995 and master degree in software engineering from University Putra Malaysia in 1998. She obtained the PhD. in Software Engineering from the same university in 2011. Currently, she is a senior lecturer at the Faculty of Computer Science and Information Technology, University Putra Malaysia. Her research interests are software metrics, software quality and software product evaluation.

**Jamilah Din** was born in 1965 at Malaysia. She received the B.Sc. in Computing Science from University of Evansville, Indiana 1987, M.Sc. in Computer Science from University Putra Malaysia (UPM) 2002 and PhD in Software Engineering from National University of Malaysia (UKM) 2009. She has more than six years of experience in industry before joining the academic field. Her research interests and work are in the areas of Object-Oriented Analysis and Design and Software Engineering. Dr. Din is now a Senior lecturer at Faculty of Computer Science and Information Technology, UPM.