MODULARITY INDEX TO QUANTIFY MODULARITY OF OPEN SOURCE SOFTWARE PROJECTS

Andi Wahju Rahardjo Emanuel¹, Khabib Mustofa²

¹Bachelor Informatics Program, Faculty of Information Technology, Maranatha Christian University
Jl. Prof. Drg. Suria Sumantri 65, Bandung, Indonesia
²Computer Science Postgraduate Program, Faculty of Math and Natural Sciences, Gadjah Mada University
Jl. Sekip Utara, Sleman, Yogyakarta, Indonesia
Email: awreman@gmail.com¹, khabib@ugm.ac.id²

ABSTRACT

In this paper a new Software Metrics called Modularity Index is proposed and analyzed for the first time. This metrics could be used as a single measure to the modularity level of Open Source Projects. Modularity has been identified by many studies as one of the key factors of the success of Open Source Projects, but how modularity should be achieved and measured systematically is not yet known. In this first attempt to identify and analyze this metrics, the correlations of this Software Metrics with other parameters currently attributed to modularity such as size, number of modules, complexity, cohesion, and coupling(fan in and fan out) are identified and analyzed. The result of the analysis is the general formula for Modularity Index in a form of matrix relation. It can be shown from the analysis that the possibility of uniting many of these attributes into this single measure called Modularity Index is highly feasible.

Keywords: Open Source Projects, Software Metrics, Modularity Index, Modularity Matrix, modularity.

1 INTRODUCTION

Open Source Projects have received many attentions recently by researchers. Once only a small time software development methodology done mainly by academics in early 1990s, this alternative to software engineering discipline is now become one of the mainstream movement in software engineering and it is attracting not only many individuals, but also big corporations such as IBM, Sun Micro system, Oracle, etc. This movement was initially started by Richard Stallman in his paper "Why software should be free" [22] and Eric Raymond in his paper "The cathedral and the bazaar" [19], and it is now become a mainstream movement which starting to threaten the proprietary software development which is mainly driven by commercial purposes. Some of the success stories about Open Source projects are the Linux Operating System, Apache Web Server, and Mozilla / Firefox web browsers, etc.

Although there have been many proofs of the success of Open Source Projects such as Linux Operating System and many of its distros, Mozilla / Firefox web browser, Apache Web Server, etc., the fact is that the number of unsuccessful and failed Open Source Projects are far more than the successful ones. For example, the Source forge Portal (sourceforge.net), one of the biggest portals for developing Open Source Software, is currently having more than 180 thousands Open Source Projects, but the number of projects that is downloaded with more than 25 thousand times are only about 5000 Projects. Some studies are trying to find the possible cause of this failures such as the lack of formal process [6], high entry barrier for new developer to contribute to the Open Source Projects [4], the poor architectural design and lack of supporting tools which is comparable to modern software development methodology [8], lack of documentation which prohibits new developers to immediately join in the projects [9], and many more.

Many researches have been conducted to measure the development of the Open Source Projects in order to discover and measure the success factors of the projects. One early important study in this area was the study by Lehman who tried to find the phases of evolution of Open Source Projects in which he found that in Open Source Project there are 7 phases of development [16]. Some other researches are relating to the size metrics such as the development of Open Source Projects relating to size, activity rate, and complexity [5], the size of the LOC and CVS commit [13]. Some researches are focusing on the quality-related metrics, for example the reuse dependencies on Debian [21], predictors of field defects on Open BSD [18], fault-proneness...
detection on Object Oriented Metrics [12], the field software usage level [2], the usage and dependency model [23]. Whereas some other researches studying to the process metrics on Open Source Projects, such as the pattern of typical commit [1], the problem resolution times [2], the patch review process [3], the patterns of download and releases with time [14], the bug dynamics [24], data mining on Sourceforge.net relating to download, activity, rank, etc. [20]. Some tools are also being developed in trying to automate the measuring process such as MUDABlue tool to measure the source code based on several categories [15], Colombus framework to measure fault proneness of Object Oriented metrics [10], Jam Tool of JFreeChart Open Source application to measure size, fan in, fan out coupling and complexity [17].

It can be observed from above researches the need to quantify modularity as a new Software Metrics. Many researches shown above which is related to software metrics for Open Source projects are process metrics that study the extrinsic characteristics of the software, which is external to the to the Open Source Project development itself, whereas modularity is focused more in the intrinsic characteristics of the Software such as the size, cohesion, coupling, fan in, fan out, etc., which are being researched in only a few studies.

2 MODEL, ANALYSIS, DESIGN, AND IMPLEMENTATION

2.1 Model

Despite all of the failures of the Open Source Projects, some studies are also trying to discover the key success factors of the Open Source Projects. Some of the important findings are the fact that modularity is one of the primary reasons behind the high quality software being produced by Open Source developers [7] [11]. Even though modularity has been identified as one of the primary factors contributing to the success of Open Source Projects, the level of modularity is still justified qualitatively by attributing it to other parameters such as size, cohesion, coupling / dependency, cohesion, fan in, and fan out. A single quantitative measure of modularity level, which is proposed to be called Modularity Index, is needed to unify all of those parameters and it may also give insight about how certain level of modularity can be achieved.

The Software Metrics will be called Modularity Index, which is intended to have the following characteristics:

- It is a form of Software Quality Metrics. Since the original purpose of this Software Metrics is to measure the modularity level of software in which the modularity has been identified in many studies as the important factor for the quality of Open Source Projects.
- This software metrics will combine several parameters currently identified as the contributing factors for modularity of Open Source Projects which are: complexity, size (LOC, function points, number of modules), coupling / dependency (fan in, fan out), and cohesion.

The value of the modularity metrics will always increasing without upper border, since some of the above parameters which correlate to the Modularity Index are also having no upper bound, such as size, complexity, number of modules, etc.

2.2 Analysis

In order to comprehend further about Open Source Projects and Software Metrics related to these projects, the theoretical background about Open Source and Software Metrics are explained in this section.

Open Source

Open Source is a software development methodology based on several distinct characteristics:

- The source code of the application is freely available for everybody to download, improve and modify. The licensing scheme of the Open Source applications such as GPL and LGPL will ensure the continual improvement of the applications by requiring everybody who improves the application to share them to everyone else.
- People who contribute to the development of the Open Source projects is usually forming a group called communities. This community will share information to each other electronically using email, mailing list, forum etc., and they are seldom or may be never meet each other face to face. The recruitment processes of the developers are completely voluntary and the hierarchies of the communities are determined by their loyalty to the project and their technical capabilities.
- The development methods of the Open Source projects are lacking of formal methodology found in commercially developed software applications. Several tools have been developed in supporting the Open Source projects such as CVS (Concurrent Versioning System), bug reporting and tracking tools, wish list, etc.
Currently, many portals have been developed as an incubator for Open Source software developers to develop and host their projects. These portals are equipped with many development tools and statistics to assist the project initiator or administrator in improving their Open Source projects and other interested contributors to join the projects. Some of the popular portals are Sourceforge.net, freshmeat.net, launchpad.net, and Google Code (code.google.com).

Software Metrics

Software metrics are defined as certain values which are expressed in some units attributed to software application. The software metrics are useful in indicates the current state of the software and enables to compare and predict the current achievement of software applications. There are several known software metrics based on its categories:

- **Size-related software metrics**: LOC (Line of Code), Memory footprint, Number of classes / headers, Number of methods, Number of attributes, Size of compiled code, etc.
- **Quality-related software metrics**: Cyclomatic complexity, Number of states, Number of bugs in LOC, Coupling metrics, Inheritance metrics, etc.
- **Process-related software metrics**: Failed builds, Defect per hour, Requirement changes, Programming time, Number of patches after release, etc.

2.3 Design

The proposed new Software Metrics is, which is proposed to be called Modularity Index, is intended to measure the modularity level of software applications quantitatively, especially for Open Source Projects. The intended characteristics of Modularity Index are stated, and the predicted correlation with other parameters currently attributed to the level of modularity such as size, number of modules, cohesion, etc. is analyzed in details.

Based on the previous identified parameters that influence modularity, these parameters have been listed with the symbols as shown in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modularity Index</td>
<td>$M$</td>
<td>Parameter indicating the quality of a software as the subject of research</td>
</tr>
<tr>
<td>Size</td>
<td>$S$</td>
<td>Size of each individual</td>
</tr>
</tbody>
</table>

2.4 Implementation

When analyzing the other parameters to this Modularity Index ($M$), the relationships are either proportional or negatively proportional depending on the observed rationale as shown in the following Table 2.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M \propto S$</td>
<td>The smaller the size of the module, the higher the tendency of the source code to become modular.</td>
</tr>
<tr>
<td>$M \propto N$</td>
<td>The larger the number of modules, it indicates the increase in granularity of the system (more modular).</td>
</tr>
<tr>
<td>$M \propto -X$</td>
<td>The more complex the application usually reflects the lower modularity of the source code.</td>
</tr>
</tbody>
</table>
The first formulation of the Modularity Index based on observations shown in Table 1 is

\[ M = -a_1 \cdot S + a_2 \cdot N - a_3 \cdot X + a_4 \cdot H + a_5 \cdot F_1 - a_6 \cdot F_0 \ldots (1) \]

Where

- \( M \) = Modularity Index
- \( S \) = Size of Module
- \( N \) = Number of Module
- \( X \) = Complexity
- \( H \) = Cohesion
- \( F_1 \) = Fan in
- \( F_0 \) = Fan out
- \( a_i \) = constants \((a_i \geq 0)\)

The six parameters contributing to the Modularity Index \((S, N, X, H, F_1, F_0)\) are not independent variables and may correlate one another. The correlation may contribute positively to one another that the combination of the two parameters should enhance the Modularity Index. The other correlation may also contribute negatively to one another that the combination of the two should lower the Modularity Index. The rationale of the relationships of each two parameters to each other and the uniformity notation for the relationship is shown in Table 3.

Combining all of those findings into a single relationship can be stated as follows:

\[ S \propto -N \propto X \propto -H \propto -F_1 \propto -F_0 \]

Converting the findings from above table, we can formulate the correlation of each parameter to one another as stated in equation 2,3,4,5,6, and 7 (note that the constant’s indexes are set to be similar to each parameter for uniformity purposes only):

Table 3. Two Parameters Relationship that Affects Modularity

<table>
<thead>
<tr>
<th>Rationale</th>
<th>Notation</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The smaller the size of each module, it will make less complex; so size is proportional to the complexity</td>
<td>( S \propto X )</td>
<td>( S \propto X )</td>
</tr>
<tr>
<td>The smaller the size of each module, the number of module increases; so number of modules is negatively proportional to the size</td>
<td>( N \propto -S )</td>
<td>( N \propto -S )</td>
</tr>
<tr>
<td>The more cohesion of the module,</td>
<td>( H \propto -S )</td>
<td>( H \propto -S )</td>
</tr>
</tbody>
</table>

The more complex, more cohesion; so cohesion is negatively proportional to complexity

\[ F_1 \propto -F_0 \]

If a module has low fan out, it call fewer or no other modules, the size is lower; so fan out is proportional to size

\[ F_0 \propto S \]

If the number of module increases, it will tend to be less complex; so number of modules is negatively proportional to complexity

\[ N \propto -X \]

If the number of module increases, the internal of the module tend to be more cohesion; so number of modules is proportional to cohesion

\[ N \propto H \]

If the number of module increases, the tendency to be called by other increases (more fan in); so the number of modules is proportional to fan in

\[ H \propto -X \]

If a module is less complex, it will tend to be more being called by other modules (high fan in); so fan in is negatively proportional to complexity

\[ F_1 \propto -X \]

If high cohesion, it will be called more by other modules (high fan in); so cohesion is proportional to fan in

\[ H \propto F_1 \]

If low fan out, it will have high fan in (tendency to be used by other modules is high); so fan in is negatively proportional to fan out

\[ F_0 \propto g_1 \cdot S - g_2 \cdot N + g_3 \cdot X - g_4 \cdot H - g_5 \cdot F_1 \]

Where

- \( b_x, d_x, e_x, f_x, g_x \) = constants \( \geq 0 \)

3 RESULT

Above formulation (from equation (1) to (7)) can be combined into a single equation to state the Modularity Index in the form of matrix relation as follows:

\[
\begin{bmatrix}
M \\
S \\
N \\
X \\
H \\
F_1 \\
F_0
\end{bmatrix} =
\begin{bmatrix}
a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\
0 & -b_1 & b_2 & -b_3 & b_4 & b_5 \\
0 & -c_1 & -c_2 & c_3 & c_4 & c_5 \\
0 & -e_1 & e_2 & -e_3 & e_4 & e_5 \\
0 & -f_1 & f_2 & f_3 & f_4 & f_5 \\
0 & -g_1 & g_2 & g_3 & g_4 & g_5 \\
0 & -h_1 & h_2 & h_3 & h_4 & h_5
\end{bmatrix}
\begin{bmatrix}
S \\
N \\
X \\
H \\
F_1 \\
F_0
\end{bmatrix}
\]

ISSN 2085-1944
Or it can be simplified as

\[ M_d = H \cdot P \]

Where each parameter are in form of matrix. Matrix H is the matrix of interest to be solved and it is proposed to be called Modularity Matrix.

These results show that the possibility uniting all the six parameters currently attributed to the modularity level of Open Source Projects called Modularity Index is highly feasible. Finding the exact values of the Modularity Matrix is the subject of the next research in finding the exact formula for Modularity Index. Selecting and justifying the correct representation regarding each of the parameters is also needed to be investigated further, for example which complexity metrics to be used, how to measure cohesion, etc.

4 CONCLUSION AND DISCUSSION

This paper proposes a new Software Metrics to measure the modularity level of Open Source Projects. Modularity is identified in several studies as one of the key factors in the success of Open Source Projects since it will promote higher quality software, but how it should be measured quantitatively and systematically is not yet known. Currently, the modularity level of a Open Source Projects is attributed qualitatively and quantitatively by six parameters such as size of module, number of module, complexity, coupling / dependency, cohesion, fan in, and fan out. The new Software Metrics attempts to unify all of these parameters into a single measure called Modularity Index.

Some of the observations and findings in this paper are:

- Modularity Index is influenced by the size of the module (S), number of modules (N), complexity (X), coupling / dependency (Fan in – Fi and Fan out – Fo), and cohesion (H)
- The preferred modular software system (high scores in Modularity Index) is: small size in each module, high number of modules in the system, low complexity, high cohesion, high fan in and low fan out.
- Each of the parameters contributing to Modularity Index are not independent to one another but it influences each other. The correlation of the new Modularity Index and other parameters may be stated in form of a matrix called Modularity Matrix.

Since this is only the first attempt in defining the Modularity Index as a new Software Metrics to quantify the level of modularity for Open Source Projects, many aspects of this Software Metrics need to be investigated and researched further, such as the exact correlation of each parameter contributing to Modularity Index to one another. The exact formulation of the software metrics should be defined more precisely in the next research, such as the assumption of linearity in the formula may also be validated. The practical method to perform measurement using this software metrics against many Open Source Projects should be realized and its result should be compared to show its effectiveness in measuring the modularity level in Open Source Projects. The current formulation of the Modularity Index will be refined based on the observed findings as the research progresses.

REFERENCES


