Abstract: Reusing existing software components is an important feature of object-oriented programming. Main advantages of it are: increase software productivity, reduce the cost, reduce testing time, reduce maintenance time and cost, improve the quality of the software. In this paper, various flaws and inconsistencies have been observed in the existing system level reuse metrics. Two new metrics namely Method Reuse Factor (MRF) and Attribute Reuse Factor (ARF) have been proposed. These metrics are helpful to find the reuse level of object-oriented systems.

Four existing system level reuse metrics, namely, Method Inheritance Factor (MIF) Attribute Inheritance Factor (AIF), Method Reuse Per Inheritance Relation (MRPIR) and Attribute Reuse Per Inheritance relation (ARPIR) have been selected for comparison with proposed metrics. Finally, metrics are evaluated against the case study.

Keywords: Reuse, Reuse Metrics, MOOD Metrics etc.

1. INTRODUCTION

Public reuse was defined by Fenton as “The proportion of a product which was constructed externally” [1]. The code defined externally means built-in-functions in various libraries. Private reuse was defined by Fenton as “the extent to which modules within a product are reused within the same product” [1]. In object-oriented term public and private reuse are not considered separately [2] due to difficulty in identifying external code. Bieman described this difficulty as due to fact that “a new system is built as an extension to an existing class library”.

In object oriented system, reuse in due to inheritance, where a class’s methods or attributes can be overridden. Direct reuse is reuse without going through an intermediate class. Indirect reuse is reuse through an intermediate class.

This research paper investigates several reuse metrics proposed by various researches and observes various flaws and inconsistencies in the existing reuse metrics. This paper proposes two new reuse metrics namely – Method Reuse Factor (MRF) and Attribute Reuse Factor (ARF).

Rest of the paper is organized in five sections. Section 2 presents a brief overview of related work and motivation of this research work. Section 3 presents the proposed metrics. Section 4 explains flaws in existing metrics and all flaws are resolved by proposed metrics. Finally, section 5 conducts the case study by taking three simple systems. Finally, section 6 presents discussion concluding remark.

2. RELATED WORK

Various efforts were put in building metrics for software reuse and reusability are seen from [4, 5]. This section gives the four metrics. These metrics are available at the project level or system level. They describe the entire project or system, not an individual class.

(a) Method Inheritance Factor (MIF): This metric is defined by Fernando Brito e Abreu [5]. MIF is defined as

$$MIF = \frac{\text{sum of all inherited methods in all classes}}{\text{sum of all inherited methods in all classes} + \text{sum of all methods declare in all classes}}$$

Higher the value of MIF more is the amount of reuse in the project. If MIF is 0%, it means there is no method inheritance in the classes.
(b) **Attribute Inheritance Factor (AIF):** This metric is also defined by Fernando Brito e Abreu [5]. AIF is defined as

\[
\text{AIF} = \frac{\text{sum of all inherited attributes in all classes}}{\text{sum of all inherited attributes in all classes} + \text{sum of all attributes declare in all classes}}
\]

Higher the value of AIF more is the amount of reuse in the project. If AIF is 0%, it means there is no attribute inheritance in the classes.

(c) **Method Reuse Per Inheritance Relation (MRPIR):** This metric is defined by Nasib S. Gill et al. [4]. MRPIR computes the total number of methods reused per inheritance relation in the inheritance hierarchy. It applies on whole inheritance hierarchy in the system. It is defined as

\[
\text{MRPIR} = \frac{\sum_{k=1}^{r} MI_k}{r}
\]

Where \( r \) = Total number of inheritance relationship

\( MI_k \) = No. of methods inherited through \( k^{th} \) inheritance relationship.

If same method is inherited through different inheritance relationships then it is computed separately in each relationship.

(d) **Attribute Reuse Per Inheritance Relation (ARPIR):** This metric is defined by Nasib S. Gill et al [4]. ARPIR computes the total number of attributes reused per inheritance relation in the inheritance hierarchy. It applies on whole inheritance hierarchy in the system. It is defined as

\[
\text{ARPIR} = \frac{\sum_{k=1}^{r} AI_k}{r}
\]

Where \( r \) = Total number of inheritance relationship

\( AI_k \) = No. of attributes inherited through \( k^{th} \) inheritance relationship.

If same attribute is inherited through different inheritance relationships then it is computed separately in each relationship.

3. **PROPOSED METRICS**

(a) **Method Reuse Factor (MRF)**

MRF is defined as

\[
\text{MRF} = \frac{\text{reuse methods in all classes}}{\text{total methods in all classes}}
\]

The MRF first counts all reuse methods in all classes or system and finally divided by the total numbers of methods in all classes or system. The MRF is project level or system level metric. In MRF, each reuse method is counts once. If same method is reused in more than one classes, then it is counted only once. In other words, If same method is inherited through different inheritance relationships then it is computed only once in all relationships. The maximum value of MRF is 1 and minimum value of MRF is 0. Method Reuse Level of system is calculated as (MRF x 100)%.

(b) **Attribute Reuse Factor (ARF)**

ARF is defined as

\[
\text{ARF} = \frac{\text{reuse attributes in all classes}}{\text{total attributes in all classes}}
\]

The ARF first counts all reuse attributes in all classes or system and finally divided by the total numbers of attributes in all classes or system. The ARF is project level or system level metric. In ARF, each reuse attribute is counts once. If same attribute is reused in more than one classes, then it is counted only once. In other words, If same attribute is inherited through different inheritance relationships then it is computed only once in all relationships. The maximum value of ARF is 1 and minimum value of ARF is 0. Attribute Reuse Level of system is calculated as (ARF x 100)%.

4. **FLAW IN EXISTING METRICS**

To study the flaws in existing metrics consider the following system:

(a) **Flaw in MIF:** Definition of the MIF is inconsistent with the 0-1 scale [6].

In the system shown in figure 1, all methods of base class have been inherited, by all classes that are able to inherit them. But MIF = 66.6%. But it seems that the MIF value for this system should be 100%.
4.4 Flaw in ARPIR
Definition of the ARPIR has same problem as discussed in MRPIR and resolved by ARF metric.

5. CASE STUDY
Consider the following three systems:

4.3 Flaw in MRPIR
Definition of the MRPIR is dependent on number of methods in the base class.
Consider the system shown in figure 1,

MRPIR = 3

If the number of methods in base class A are 4 then MRPIR = 4. That means value of MRPIR depends upon the number of inherited methods in the base class. There is further no maximum and minimum value of MRPIR.

Proposed Metric MRF resolved this problem. In the system shown in figure 1, value of

MRF = 1 i.e. 100%.

If the number of methods in base class A are 4 then also value of MRF = 1 i.e. 100%.
Table 1 shows the value of MIF, MRPIR and MRF for system 1, system 2 and system 3.

<table>
<thead>
<tr>
<th>SYSTEM 1</th>
<th>SYSTEM 2</th>
<th>SYSTEM 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIF</td>
<td>0.545</td>
<td>0.533</td>
</tr>
<tr>
<td>MRPIR</td>
<td>3</td>
<td>2.66</td>
</tr>
<tr>
<td>MRF</td>
<td>0.8</td>
<td>0.857</td>
</tr>
</tbody>
</table>

A comparison study of MIF, MRPIR and MRF is shown in following bar chart:

Table 2 shown the value of AIF, ARPIR and ARF for system 1, system 2 and system 3.

<table>
<thead>
<tr>
<th>SYSTEM 1</th>
<th>SYSTEM 2</th>
<th>SYSTEM 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIF</td>
<td>0.286</td>
<td>0.333</td>
</tr>
<tr>
<td>ARPIR</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ARF</td>
<td>0.4</td>
<td>0.5</td>
</tr>
</tbody>
</table>

A comparison study of AIF, ARPIR and ARF is shown in following chart:

6. DISCUSSION AND CONCLUDING REMARK

According to table 1, value of MRF is maximum is system 2. It indicates that level of method reuse is highest in system 2 as compare to system 1 and system 3. However, value of MIF is maximum in system 3. It indicates that level of method reuse is highest in system 3 as compare to system 1 and system 2. The value of MRPIR is same in system 1 and system 3. It indicate that level of method reuse is same in system 1 and system 2.

According to table 2, value of ARF is maximum in system 3. It indicates that level of attribute reuse is highest in system 3 as compare to system 1 and system 2. However, value of AIF is also maximum in system 3. The value of ARPIR is same in system 1, system 2 and system 3. It indicates that level of attribute reuse is same in all systems.

This paper have been proposed two new system level reuse metrics. These metrics resolved the flaws or inconsistencies in the existing reuse metrics. These metrics are helpful to find the reuse
level of object-oriented system. This study is supported by simple case study however, the same study can be replicated empirically with large industrial projects to generalize results.

REFERENCES


