CHAPTER 3

OBJECT ORIENTED REUSE METRICS

3.1 INTRODUCTION

In this chapter, we discuss about two new object oriented reuse metrics namely Method Reuse Factor (MRF) and Attribute Reuse Factor with Inheritance (ARFI) that are proposed in this research work to provide quantitative measurement for reuse in product or process. The major advantage of these proposed metrics is that they help in formulating and calculating object oriented design function points more precisely and also helps in estimating procedures to make accurate quantification.

The proposed MRF and ARFI have been validated by considering two case studies where the first one Constructive Cost Model (COCOMO) estimation (Boehm Barry 1981) and the other is the enhanced "OO Design Function Points Counting Method" proposed in this research work by extending the OODFP developed by Janaki Ram et al (2000).

The COCOMO Model proposed by Boehm is the most widely used software estimation model (Boehm 1981). The "COCOMO" model predicts the effort and duration of project, based on inputs relating to the size of the resulting systems and the number of cost drivers that effect productivity. Most of the COCOMO results including the estimates for the requirements and maintains are derived from the equation.
\[ PM - Cx(KDSI)^n \]  

where  
- \( PM \) - number of person-months 
- \( C \) - constant 
- \( KDSI \) - thousands of delivered source instructions 
- \( n \) - constant

In any COCOMO model, the most important factor contributing to project duration and cost are based on the development modes. Basically there are three development modes namely Organic - mode, Semi detached mode and Embedded mode.

Parameters like size and cost are very much needed to utilize the resources of software projects effectively. There are a number of methods available to calculate the size like lines of code, function points, etc. As these lines of code are language dependent (Levitin 1996), function point counting procedure has been chosen to estimate the functionality of software from user point of view. This measure is considered to be independent of implementation.

The traditional functional point counting procedure is not suitable for measuring the functionality of object oriented systems because the raw functionality of software, communication between objects and lack of inheritance (Jalote 1998). For estimating the size of OO software, the criteria is based on the adaptation of classical function point method to object oriented software to map the concept of functions to methods in OO software. Most elements in the object models are related to application domains so the perspective is still from that of the user. At the OO design phase, the object model reflects the implementation choices which include the aspect of the
system that are not specified in the required documents. The count of Object Oriented Function Point (OOFP) includes the functionality and measurement perspective of the designer. To map the function point in OO software, logical files are mapped on to the class and transactions are mapped on to their methods. A logical file in the function point approach is a collection of related user-identifiable data. A class in an object model encapsulates a collection of data items and objects that are instances of class in OO world corresponding to records of a logical file in data processing applications.

In the past, a method for estimating the function point from designer perspective has been developed and estimated based on functionality of OO software and complexity of each class (Janakiram et al 2000). Various methods have been employed by various researchers to enhance various aspects of software products such as quality, reusability etc.

Reusability is the degree to which things can be reused. In order to improvise the project estimation and measurement techniques improvised i.e. to make more accurate estimation to evaluate size and cost of software project and there by enhancing the performance, object oriented approach is well suited. The reuse is a direction towards measuring responsibility and in this, metric provides an approach to measure method reuse factor and attribute reuse-factor with and without inheritance. In this research work, two case studies have been considered to validate the method reuse-factor and attribute reuse - factor with inheritance, which can be used in the project estimation and in improving the counting method for the estimation of object oriented design function points. Moreover, the new set of object oriented reuse metrics proposed can be used to assess reuse in both process and products. Thus, Method Reuse Factor, Attribute Reuse Factor, Method Reuse Factor with Inheritance (MRFI) and Attribute Reuse Factor with Inheritance (ARFI) have
been defined. However, the main focus of this chapter is to study the impact of the use of MRF and ARFI.

3.2 COCOMO – MODEL

The development models of the model are as follows.

**Organic mode**

Relatively small, simple projects in which small teams with good application and experience, work to a set of less than rigid requirements.

**Semi detached mode**

Intermediate size and complexity project in which teams with mixed experience must meet with a mix of rigid and less than rigid requirements.

**Embedded mode**

Software projects that must be developed within a set of tight hardware, software and operational constraints. All these are focused only on procedure oriented software. However, the predictive object points method was proposed by Teologu (1999), based on hierarchy and weighted methods per class and hence is suitable for OO systems in which methods are weighted according to the constructor, selector, modifier, and integrators.

3.3 DEFINITION OF METHOD REUSE FACTOR (MRF)

In order to improve the accuracy in estimation a new metric, Method reuse factor is introduced which is defined as the ratio of methods used to methods available in a class.
Method Reuse Factor = \sum_{i=1}^{TC} \sum_{j=1}^{m} \frac{Methods used}{Methods available} \quad (3.2)

Where

\( Tc \) - total number of classes
\( m \) - total methods in a class

The value of this Method Reuse Factor always lies in the range of 0 to 1.

Different projects with different KLOC are considered to make the estimation of effort, time and cost. These parameters are calculated and applied to COCOMO model with and without using Method Reuse Factor and the results obtained are shown.

3.4 APPLICATION OF MRF IN COCOMO MODEL

Equations (3.4) and (3.5) provide the set of basic equations that are used to calculate the effort, time and cost for all the three modes of COCOMO Model. The results obtained are shown in Table 3.1 for different projects of different sizes and complexity.

\[
\text{Effort} = a_1 \times (\text{KLOC})^{a_2} \quad (3.4)
\]

\[
T\text{dev} = b_1 \times (\text{Effort})^{b_2} \quad (3.5)
\]

Where
- KLOC is the estimated kilo lines of code,
- \( a_1, a_2, b_1, b_2 \) are constants for different categories of software products,
- Tdev is the estimated time to develop the software in months,
• Effort is the total development effort required to produce the software product, in programmer-months (PMs).

3.4.1 Application of MRF and Analysis

Two sets of data pertaining to this analysis are taken to evaluate this approach. First data set is synthetic data and the second set is collected from a CMM Level 5 Company. The next section shows the results obtained and analysis.

3.4.1.1 Effort estimation using COCOMO model with and without MRF for synthetic data

Analysis 1

In the first case of synthetic data, 100 projects having different KLOC in ascending order magnitude are considered as given in Table A 1.1 (Appendix 1). After assuming KLOC and MRF, the effort has been estimated with and without MRF in three different modes of COCOMO i.e Organic, Semidetached and Embedded Mode. From the experiments carried out with synthetic data, the optimal value for MRF is obtained as 0.8. The same is depicted in the form of graph and are shown in Figures 3.1, 3.2 and 3.3, which highlight the advantage of improved estimation of thses parameters while using MRF in the COCOMO Model.
Figure 3.1 Effort Comparison Organic – Synthetic Data

Figure 3.2 Effort Comparison Semidetached – Synthetic Data
Figure 3.3 Effort Comparison Embedded-Synthetic Data

3.4.1.2 Effort estimation organic, semidetached and embedded mode with and without MRF-industrial data

Analysis 2

In the second case of data set, the 96 industrial projects having different KLOC in various magnitudes are given in Table A1.2 (Appendix 1). After collecting KLOC and computing MRF, the effort has been estimated with and without MRF in three different modes of COCOMO model that is Organic, Semidetached and Embedded Mode (Table A1.2). The same is depicted in the form of graph and are shown in the following Figures 3.4, 3.5 and 3.6, which highlight the advantage of improved estimation of theses parameters while using MRF in all the three modes.
Figure 3.4 Effort Comparison Organic – Industrial Data

Figure 3.5 Effort Comparison Semidetached – Industrial Data
3.4.1.3 Time estimation organic, semidetached and embedded mode with and without MRF-synthetic data

Analysis 3

In the first case of synthetic data, the 100 projects having different KLOC in ascending order magnitude are given in Table A 1.3 (Appendix 1). After assuming KLOC and MRF, the time has been estimated with and without MRF in three different modes of COCOMO and are shown in Table A 1.3. The value of MRF is computed as 0.8. The same is depicted in the form of graph and are shown in the following Figures 3.7, 3.8 and 3.9, which highlights the advantage of improved estimation of these parameters while using MRF in all the three modes of COCOMO Model.
Figure 3.7 Time Comparison Organic Mode – Synthetic Data

Figure 3.8 Time Comparison Semidetached Mode – Synthetic Data
3.4.1.4 Time estimation organic, semidetached and embedded mode with and without MRF - industrial data

Analysis 4

In case of time estimation in all the three modes, industrial data in KLOC has been shown in Table A 1.4 (Appendix 1). After collecting KLOC and computing MRF, the time has been estimated with and without MRF in three different modes of COCOMO that is Organic, Semidetached and Embedded Mode. The same is depicted in the form of graph and are shown in the following Figures 3.10, 3.11 and 3.12, which highlights the advantage of improved estimation of theses parameters while using MRF in all the three mode of COCOMO Model.
Figure 3.10 Time Comparison Organic Mode – Industrial Data

Figure 3.11 Time Comparison Semidetached Mode – Industrial Data
3.4.1.5 Cost estimation Organic, Semidetached and Embedded Mode with and without MRF - Synthetic Data

Analysis 5

In case of cost estimation for the three modes same synthetic data of 100 projects was considered with 0.8 MRF and the same is shown in Table A 1.5 (Appendix 1). The same is depicted in the form of graph and are shown in the following Figures 3.13, 3.14 and 3.15, which highlights the advantage of improved estimation of theses parameters while using MRF in all the three modes of COCOMO Model.
Figure 3.13 Cost Comparison Organic Mode – Synthetic Data

Figure 3.14 Cost Comparison Semi – Detached Mode – Assumed Data
3.4.1.6 Cost estimation Organic, Semidetached and Embedded Mode with and without MRF - Industrial Data

Analysis 6

While considering the cost estimation with industrial data same set of 96 projects which were used for time and effort estimation have been considered and computed the value of MRF and the same is shown in Table A 1.6 (Appendix 1) for all the three modes. The same is depicted in the form of graph and are shown in the following Figures 3.16, 3.17 and 3.18, which highlights the advantage of improved estimation of theses parameters while using MRF in all the three mode of COCOMO Model.
Figure 3.16 Cost Comparison Organic Mode – Industrial Data

Figure 3.17 Cost Comparison Semidetached Mode – Industrial Data
From the results we infer that the use of MRF provide better performance in comparison with the basic COCOMO model.

### 3.4.1.7 Discussion

In all the above mentioned analysis, for every one of the models, the trend line for "With MRF" and "Without MRF" have been determined. The slopes of each one of the trend lines for both synthetic and Industry data are determined and tabulated in Tables 3.1 and 3.2.

#### Table 3.1 Trend Line Comparison for Synthetic Data Set

<table>
<thead>
<tr>
<th>Modes</th>
<th>Organic</th>
<th>Semidetached</th>
<th>Embedded</th>
<th>Organic</th>
<th>Semidetached</th>
<th>Embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort</td>
<td>79.80383</td>
<td>84.04061</td>
<td>86.78431</td>
<td>71.39144</td>
<td>79.23554</td>
<td>83.8863</td>
</tr>
<tr>
<td>Cost</td>
<td>88.65191</td>
<td>88.63587</td>
<td>88.60533</td>
<td>87.60288</td>
<td>87.74256</td>
<td>87.71284</td>
</tr>
</tbody>
</table>

#### Table 3.2 Trend Line Comparison for Industrial Data Set

Figure 3.18 Cost Comparison Embedded Mode - Industrial Data
Without MRF (- in degrees) | With MRF (- in degrees)
---|---
Modes | Organic | Semidetached | Embedded | Organic | Semidetached | Embedded
Effort | 76.13014 | 78.03777 | 81.54816 | 66.86038 | 75.96358 | 76.51448
Time | 78.03777 | 76.22822 | 71.84713 | 67.46443 | 68.19843 | 67.29492
Cost | 81.31939 | 80.41198 | 80.83746 | 75.14411 | 77.73504 | 75.25626

The above table exhibits the slow increase in the trend-lines for each one of the 18 Models. It is to be noted that the slopes are indicated by 9 and 0 respectively for "without MRF" and "with MRF". The trend-line for each model exhibits that the slope pertaining to "without MRF" is always greater than that of "with MRF". This trend rapidly goes up beyond more than 100 KLOC. The reason for rapid increase in the slope accounted mainly due to increase in the KLOC for different set of projects. By applying the new metric called MRF, it is possible to reduce the KLOC.

3.5 ATTRIBUTE REUSE FACTOR WITH INHERITANCE

Another new metric namely (ARFI) Attributed Reuse Factor with Inheritance is proposed by considering the inheritance of class at system level. This factor gives exactly how many data items are used in the derived class. In case of the OODFP counting procedure, Janaki Ram et al (2000) considered the actual number of data items that are used in the derived class. In their work, the class complexity was used to improve the OO design.

In this thesis, we incorporated attribute reuse factor in the inherited class to improve the counting procedure. By using this, one can make accurate estimate and there by, size and cost of the software projects can be better evaluated. This information at the design phase is very crucial and very much
useful for the designer as well as the developer.

**Definition of ARFI**

In addition to the MRF a new metric called ARFI is introduced to calculate and improvise the counting procedures of OO design which is defined by the formula

Attribute Reuse Factor

\[
ARFI = \sum \sum \frac{\text{Number of Inherited attributes}}{\text{Inherited attributes + class attributes}}
\]  

(3.3)

Where

- \( Tc \) - total number of classes
- \( a \) - total attributes in class

The value of this Attribute Reuse Factor also lies in the range of 0-1

**3.6 APPLICATION OF ATTRIBUTE REUSE FACTOR WITH INHERITANCE**

The newly introduced Attribute Reuse Factor with Inheritance is computed based on the equations and are incorporated in the counting procedures of function point estimation.

Let \( C_b \) be the number of base classes having \( D \) number of DETs (Data Element Type) and \( R \) number of RETs (Record Element Type). Let \( C_d \) be the number of derived class as derived from base class \( C_b \) having its own data elements numbering \( D_{ce} \).
Total elements considered in the derived class = All the data items of base class + its own elements of derived class

\[ = D + R + D_{ce} \quad (3.4) \]

Total elements considered in the derived class by incorporating ARF = (All the data items of base class) *ARF+ its own elements of derived class

\[ = ((D+R)\times ARF) + D_{ce} \quad (3.5) \]

IFPUG (1999) states that the KLOC can be computed using Unadjusted Object Oriented Design Function Points (UOODFPO). KLOC as a measure of project size is estimated in both the above cases using the following equation as in IFPUG.

\[ \text{Estimated KLOC} = \text{UOODFP} \times \text{CCV} \times 30 \quad (3.6) \]

where CCV refers to Class Complexity Value constant 30 is dependent on the programming language used.

The KLOC obtained from the above equation is compared with the actual KLOC of any project. The comparison is represented as percentage error projections in a tabular form as follows.

\[ \% \text{ error} = (\text{Actual KLOC} - \text{Estimated KLOC}) / \text{Actual KLOC} \quad (3.7) \]

Hundred different projects of different size and complexity are taken up for validating the newly introduced ARFI. The results obtained are tabulated in Table 3.3 in the form of percentage error.
Table 3.3 Percentage Error with and without ARFI

<table>
<thead>
<tr>
<th>SI. No</th>
<th>Projects Number</th>
<th>% of Error with ARF</th>
<th>% of Error without ARF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>6.3</td>
<td>31.2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>15.5</td>
<td>29.2</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>16.1</td>
<td>25.6</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>29</td>
<td>34.8</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>11.2</td>
<td>19.3</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>17.6</td>
<td>18.6</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>23</td>
<td>39.6</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>25.8</td>
<td>36.4</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>21.6</td>
<td>31</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>31.4</td>
<td>36.7</td>
</tr>
<tr>
<td>11</td>
<td>55</td>
<td>9.7</td>
<td>25.1</td>
</tr>
<tr>
<td>12</td>
<td>60</td>
<td>19.6</td>
<td>23.8</td>
</tr>
<tr>
<td>13</td>
<td>65</td>
<td>6.1</td>
<td>10.1</td>
</tr>
<tr>
<td>14</td>
<td>70</td>
<td>10.9</td>
<td>12.7</td>
</tr>
<tr>
<td>15</td>
<td>75</td>
<td>4.6</td>
<td>9.5</td>
</tr>
<tr>
<td>16</td>
<td>80</td>
<td>6.1</td>
<td>8.9</td>
</tr>
<tr>
<td>17</td>
<td>85</td>
<td>5.1</td>
<td>9.3</td>
</tr>
<tr>
<td>18</td>
<td>90</td>
<td>4.9</td>
<td>8.1</td>
</tr>
<tr>
<td>19</td>
<td>95</td>
<td>3.6</td>
<td>5.7</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>2.8</td>
<td>5.1</td>
</tr>
</tbody>
</table>
Based on the two graphs shown in Figure 3.19 exhibits a curvi-linear trend. In so far as the model pertaining to "with ARFI" exhibit a uni-model (Peak) trend. Where as the model pertaining to "without ARFI" exhibits two lower depression occur to the left and to the right of the middle number of projects. The actual parabolic fitting curve for "with ARFI" model is taken as $(x-a)^2 = X (y - (3))$. Similarly without ARFI model the curve fitting is based upon $(y - 5) = \forall (x-y)$, where $a$, $p$ are the peak and $y$, $8$ are the depression points.

3.7 CONCLUSION

This chapter proposed a modification to the COCOMO estimation. In this thesis, an attempt has been made to modify estimation so that the performance of COCOMO is improved by defining and incorporating Method Reuse Factor, which improved the cost estimation. In all the three modes of COCOMO, the application of Method Reuse Factor has improved the original estimates of schedule time and cast to the tune of 26 percent. The variation
observed from organic to embedded is 20.5 to 23.5 percent in effort, 8.252 to 8.2 percent in time, and almost retained constant (26.6 to 26.8) in the cost over the conventional COCOMO empirical estimates.

Though there is relatively less impact on smaller size projects, as the size of the project increases, for larger project sizes the impact of Method Reuse Factor on effort, time and cost are significant.

This chapter also proposed another metric for OODFP estimation method from designer's perspective. The main advantage of ARFI proposed in this work is that it helped to improve the accuracy of estimation. Moreover, this metric is applied at the design phase enabling a better design so that the design can be modified to get an optimal design.