Statistical Analysis of Cohesion metrics

H.B.Vincentraj  
Department of Computer Science  
Bishop Heber College  
Trichy-17, India  
Email:hbvincent_raj@hotmail.com

S.Hari Ganesh  
Department of Computer Science  
H.H. The Rajah’s College  
Pudukottai -1, India  
Email: hariganesh17@gmail.com

Abstract: One of the most prevalent programming languages in software development environment is object oriented programming (OOP). Though the prediction of maintenance efforts in OO programming is under the scope of software metrics, the problem incurred with software maintenance is not completely evacuated, as most of the metrics are not statistically validated to prove their usability. The objective of this paper is to statistically validate the three software cohesion metrics namely Co- incidental Functional Cohesion Metric (CFCOM), Sequential Cohesion Metric (SCOM), Communicational Cohesion Metric (CCOM) that are designed to predict the maintenance effort of the software. The paper also investigates the relationship between the module complexity and maintenance time. The results are clearly shown that the proposed cohesion metrics helps in the prediction of software complexity for maintenance.

Keywords: cohesion, maintenance effort, SCOM, CCOM, CFCOM

Introduction

The features of object oriented programming have leaded the increased implementations of OO software in development environment. Besides the enormous features, software maintenance is one of the most important and vital feature of OOP that allow the programmers to change or extend new modules in the existing software. Thus expecting the software to be created in such a way that it is dynamic and vulnerable for future modification. Creating maintainable software still remains a challenge in OOP particularity when the design of the modules is complex [1]. Software complexity metrics act as a good indicator to measure the complexity of the software where a high metric value may indicate high complexity, low metric value may indicate low complexity and vice versa [2][3][4]. Several studies have been conducted to investigate the relationship between design complexity and maintenance performance with which most of the findings have concluded that software metrics are used to predict the maintenance performance of the software. Thus, the objective of this study is to prove whether the proposed software complexity metrics CFCOM, SCOM and CCOM contribute to the prediction of maintenance performance.

Review of Literature

Edson et al. [5] have presented two metrics namely complexity PLA and extensibility PLA for software Product Line Architecture (PLA) and concluded that the metrics are relevant indicators of complexity and extensibility of PLA by presenting correlation analysis.

Thamburaj et al. [6] have validated a software complexity metric named Cognitive Weighted Attribute Hiding Factor (CWAHF) through correlation analysis and claimed that the metric is robust and accurate in predicting the complexity of the software.

Aloysius et al. [7] have empirically explored the validation of three cognitive complexity metrics such as Attributed Weighed Class Complexity (AWCC), Cognitive Weighted Response for a Class (CWRFC) and Cognitive Weighted Coupling between Object (CWCBO). The authors have also investigated that the effect of design complexity on maintenance time and found that the experiment is useful in predicting maintenance performance.

SALMAN et al. [8] have introduced three component based complexity metrics namely Total Number of Components (TNC), Average Number of Methods per Component (ANMC) and Total Number if Implemented Components (TNIC). The authors have conducted a case study to detect the power of complexity metrics in predicting integration and maintenance efforts and claimed the component complexity metrics are useful in predicting the integration and maintenance efforts.

Rajnish [9] has presented a complexity metric named Attribute Method Complexity (AMC) to find whether the classes of software is less, moderate or high complex. The author has theoretically evaluated against Weyuker’s properties and experimentally validated against three open source system and the results indicates that the new complexity metric is correlated well with existing complexity metrics and used as predictors of complexity of class.
Statistical Design.
The statistical design of the study suggests that the module design, maintenance task, and programmer knowledge may influence maintenance performance. Thus, the maintenance performance is considered as the dependent variable and the module design, maintenance task and the programmers knowledge are the independent variables to find the relationship to predict maintenance effort. The empirical study is conducted as an experiment over the students of Bharathidasan university, India.

Description of Dependent Variable
Software maintainability refers to the ease with understanding or modifying software programs or modules which can be defined from the time taken to comprehend, modify, develop a software code or to derive the accuracy of the results [10]. The objective of this study is to measure the time factors with comprehending, modifying and developing the existing programs for verifying the underlying relationship with maintenance effort. The pictorial representation of the empirical design is shown in Figure. 1.

Description of Independent Variables
Module Design
Co-incidental Functional Cohesion Metric (CFCOM) [11], Sequential Cohesion Metric (SCOM) [12], Communicational Cohesion Metric (CCOM) [13] are the three existing cohesion metrics that identifies the quotient of cohesion types that exists in the software. The description of each metric is given below.

Co-incidental Functional Cohesion Metric (CFCOM)
The proposed Coincidental- Functional Cohesion Metric is a novel metric intended to assess whether the given module or class is coincidentally cohesive or functionally cohesive. As it is discussed earlier a class or module should be functionally cohesive as it reduces the complexity in understanding, comprehending and maintaining the program. On the other hand, a coincidental cohesive class or module rather increases the complexity as a whole.

CFCOM can be calculated by using the equations as follows:

\[
CFCOM = \frac{\sum \cap \cap_{i=1}^{n} AM_{i} \cap TAC}{TAC}
\]

Here, AM\textsubscript{i} refers to the total number of attributes used in method \textit{i} and TAC refers to the total number of Attributes defined in the class. The metric CFCOM intersects the variables of methods in a class with the variables that are defined in a class. The summation of the intersected variables is then divided by the overall possibilities that could be made within a class. Value 1 of CFCOM represents that the class is functionally cohesive, and Value 0 of CFCOM represents the class as coincidentally cohesive. Coincidentally cohesive class is an alarm for the programmers to redefine the class into an inseparable unit.

Sequential Cohesion Metric (SCOM)
Sequential cohesion refers to the communication between two methods where the output of one serves as an input to the other in a sequence of method calls. Software with sequential cohesion is accepted as it increases the possibility...
of integration of elements of within the methods of a module. So far, there is no such metric is proposed to calibrate the level of sequential cohesion presented in a module. Hence, as a novel attempt, we have proposed a sequential cohesion metric that evaluates the percentage of sequential cohesion involved in a module.

\[
SCOM = \frac{\sum_{i=1}^{n} m_i \cap m_{i+1}}{(n-1) \times TAC} \times 100\%
\]

(1)

‘n’ denotes the total number of methods in the module, ‘mi’ denotes ith method whereas \( m_i \cap m_{i+1} \) is the intersection of attributes of mi and mi+1, and TAC refers to the total number of attributes in a class. SCOM is the percentage of the summation of intersected variables of two consequent methods divided by the possible sequential cohesion. A software possessing 100% of SCOM denotes a strong sequential cohesion and 0% denotes weak sequential cohesion. The implementation of sequential cohesion in software enhances the modularity of software program.

**Communicational Cohesion Metric (CCOM)**

Communicational Cohesion is the grouping up of methods that operate on the same data within a class or module for measuring the integrity of methods. Software with high quotient of communicational cohesion ensures a good representation of class design that proves the increased integrity of methods within a module or class. Software metric that evaluates the level of communicational cohesion in software modules is a still being considered as a thrust area in research which is yet to be focused. Hence, in this paper an attempt is made to propose a communicational cohesion metric (CCOM) for assessing the percentage wise communicational cohesion that the software modules are designed with. The low level communicational cohesion suggests developers for the modification of software code by increasing the sharing of attributes within the methods of class or modules. The CCOM value of a module is the percentage fraction of sum of intersecting variables between methods by both sums of intersecting and non-intersecting variables between the methods which is denoted using formula.

\[
CCOM = \frac{CM}{CM + NIVBM} \times 100\%
\]

CM is the communicational measure which is derived by multiplying the sum of intersecting variables between methods by two and can be represented using the formula shown in Equation.

\[
CM = 2 \times IVBM
\]

IVBM represents the sum of Intersection of Variables Between Methods which is denoted using the formula for the computation of IVBM.

\[
IVBM = \sum_{i=1}^{n} m_i \cap m_j
\]

where ‘n’ denotes the total number of methods in the module, ‘mi’ and ‘mj’ denotes i\(^{th}\) and j\(^{th}\) methods whereas \( m_i \cap m_j \) is the intersection of attributes of \( m_i \) and \( m_j \). Finally, NIVBM represents the sum Non-intersecting of Variables Between Methods which is depicted in Equation.

\[
NIVBM = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (m_i \cap m_j)
\]

A software module with the CCOM 100% value denotes a strong communicational cohesion and 0% value denotes weak communicational cohesion. The implementation communicational cohesion in software enhances the modularity of software program.

**Maintenance Task**

Maintenance task is the second independent variable of the study which can be majorly classified into corrective, perfective and adaptive [14]. Corrective maintenance refers to the activity of debugging when the system is in operation. This task is error-driven and is required throughout the life of the software, as software is never truly error free. Perfective maintenance refers to the modifications of software that meets user needs. In this study the perfective maintenance is referred as the comprehensive maintenance task. Finally, adaptive refers to the change of compiler, operating systems or documents and hardware peripherals. In this study, the corrective and comprehensive maintenance tasks are used.

**Programmer Knowledge**

The third independent variable of the study is the knowledge of the programmers, as the maintenance of software requires adequate knowledge in the language by programmers.
Statistical Analysis
The following research queries have been set for deriving the hypothesis of the study
Q1: Is there a relationship between the complexity of module design with the maintenance time?
Despite the availability of numerous statistical methods to examine the relationship between the dependent and independent variables, this study employs the ANOVA, Correlation and Linear regression techniques to see whether the complexity metric can be used as an indicator for predicting the maintenance efforts of the software. The research test is expressed in terms of Null (H_N) and Alternate hypothesis (H_A).
H_N1: There is no difference in the maintenance effort with respect to complexity metrics
H_A1: There is a difference in the maintenance effort with respect to complexity metrics
H_N2: There is no correlation between maintenance effort and complexity metrics
H_A2: There is a correlation between maintenance effort and complexity metrics
H_N3: There is no linear relationship between maintenance effort and complexity metrics
H_A3: There is a linear relationship between maintenance effort and complexity metrics
The complexity of the system is measured with the three cohesion metrics CFCOM, SCOM and CCOM by applying the research hypothesis to each of the metrics.

Experimental Study
The experiment for the maintenance effort prediction is conducted over sixty students from two sections studying undergraduate and postgraduate programs. The samples are expected to have OO experience as a prerequisite. The OO and the total programming experience of the students ranges from six months to six years. A questionnaire is prepared to collect the students data such as name, roll no, department, programming experience and OO experience. The students have mean of 43.56666667 months of programming experience with 15.4395236 of standard deviation and have a mean of 15.65 months of OO experience with 4.784784913 of standard deviation. The experiment is conducted with two independent programs involving corrective and comprehensive maintenance tasks. The programs are designed with two versions of complexities such as low and high. The samples are asked to work either with low or high complexity version of each program and to report the maintenance time. Table 1 depicts the allocation of programs to samples.

Table 1. Program Allocation to Samples

<table>
<thead>
<tr>
<th>Complexity Version</th>
<th>Comprehensive (Company)</th>
<th>Corrective (test)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Complexity</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>High Complexity</td>
<td>40</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>60</td>
<td>120</td>
</tr>
</tbody>
</table>

The programs are assigned by letting the students to attempt the low version of one program and the higher version of the other. The name of the programs is company and test. Comprehensive task involves in the derivation output to the program and the second corrective maintenance task is to change the functionality of the system. The metric values for the corresponding programs are given in Table 2.

Table 2. Cohesion metrics Values of complexity versions

<table>
<thead>
<tr>
<th></th>
<th>Company (comprehensive)</th>
<th>Test (Corrective)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low Complexity</td>
<td>High Complexity</td>
</tr>
<tr>
<td>CFCOM</td>
<td>0.956</td>
<td>0.173</td>
</tr>
<tr>
<td>SCOM</td>
<td>0.824</td>
<td>0.211</td>
</tr>
<tr>
<td>CCOM</td>
<td>0.912</td>
<td>0.146</td>
</tr>
</tbody>
</table>

Results of the Statistical Analysis
The aim of this experiment is to conduct the validation of the three cohesion complexity metrics by evaluating the ability to predict average maintenance time.
ANOVA
To start with, the ANOVA (Analysis of Variance) test is conducted to analyze the average maintenance time within and between the programmer groups.

Table 3. ANOVA for Average maintenance time Vs Complexity versions

<table>
<thead>
<tr>
<th>Program</th>
<th>Average Maintenance Time</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low Complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td>14.65</td>
<td>24.13591128</td>
<td>&lt;0.00001</td>
</tr>
<tr>
<td>Test</td>
<td>7.975</td>
<td>27.034137</td>
<td>&lt;0.00001</td>
</tr>
<tr>
<td></td>
<td>High Complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td>21.725</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>27.95</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3 shows the average maintenance time for the company program, where the high complexity version of the program has higher mean time of 21.75 minutes than the lower complexity program 14.65 minutes. Since, the P-value is less than 0.00001 the probability of assuming the null hypothesis is less than 0.00001. Thus, it has been proved that complexity of the software affects the average maintenance time and concluded that the higher metric value of CFCOM, SCOM and CCOM signifies the lower complexity value and the lower metric values of CFCOM, SCOM and CCOM signifies high complexity. The pictorial representation of Time analysis is shown in Figure 2.

Regression Analysis
A regression analysis is conducted to elucidate the relationship between metrics and average maintenance time and the results are shown in Table 4, which depicts the cohesion metrics values of low and high complexity programs of maintenance tasks and the average maintenance time.

Table 4. Regression Analysis of Cohesion metrics with Average Maintenance Time

<table>
<thead>
<tr>
<th>Maintenance Task</th>
<th>Complexity Version</th>
<th>CFCOM</th>
<th>SCOM</th>
<th>CCOM</th>
<th>Average Maintenance Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehensive</td>
<td>Low</td>
<td>0.956</td>
<td>0.824</td>
<td>0.912</td>
<td>14.65</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>0.173</td>
<td>0.211</td>
<td>0.146</td>
<td>21.725</td>
</tr>
<tr>
<td>Corrective</td>
<td>Low</td>
<td>0.869</td>
<td>0.751</td>
<td>0.964</td>
<td>7.975</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>0.314</td>
<td>0.187</td>
<td>0.139</td>
<td>27.95</td>
</tr>
</tbody>
</table>

Table 5 shows the results of the multiple linear regression. The correlation between the independent and dependent variable Multiple R (0.99) denotes a strong linear relationship between them. The coefficient of determination R2 (0.997) determines close to 99% of the variation in the dependent variable is explained by independent variables. The Adjusted R Square (0.99) adjusts the number of terms in a model, and increases only when meaningful values are added. Standard Error (0.256063) depicts the standard deviation error. The significance of F (0013) explains there is only 1% of chance that the regression output is merely a chance occurrence. Moreover, the p-value of the
independent variables are not greater than 0.05. Hence, the null hypothesis is rejected for the multiple linear regression.

Table 5. Regression Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple R</td>
<td>0.99</td>
</tr>
<tr>
<td>R Square</td>
<td>0.997</td>
</tr>
<tr>
<td>Adjusted R Square</td>
<td>0.99</td>
</tr>
<tr>
<td>Standard Error</td>
<td>0.256063</td>
</tr>
<tr>
<td>Significance F</td>
<td>0.0013</td>
</tr>
</tbody>
</table>

From the results of Table it is clearly understood that the value of the cohesion metrics clearly describes the complexity of the software code. Low cohesion complexity metric values in CFCOM, SCOM and CCOM denote that the software is highly complex and maximizes the maintenance cost. High cohesion complexity metric values in CFCOM, SCOM and CCOM denote that the software is less complex and minimize the maintenance cost.

Conclusion

This paper aims at statistically exploring the validation of the three OO cohesion metrics: Co- incidental Functional Cohesion Metric (CFCOM), Sequential Cohesion Metric (SCOM), and Communicational Cohesion Metric (CCOM). A planned experimentation is conducted in the laboratory for achieving the research objectives of the paper. The statistical tests ANOVA, regression and correlation quantitatively analyze the experimental data and the results have proven that the proposed cohesion metrics are found to be useful in predicting the complexity of the software by rejecting the null hypothesis.

References