Volume 3, Issue 11, November 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Maintenance Effort Prediction Model Using Cognitive Complexity Metrics

Dr. A. Aloysius[1]
[1]Assitant Professor in the Dept of Computer Science,
St. Joseph's College, Tiruchirappalli – 620002, India

Dr. L.Arockiam[2]
[2]Associate Professor Dept of Computer Science,
St. Joseph's College, Tiruchirappalli – 620002, India

Abstract: ***The Object-Oriented (OO) technology is becoming an increasingly popular software development environment. Researchers agree that, although maintenance may turn out to be easier for OO systems, it is unlikely that the maintenance burden will completely disappear. One approach to controlling software maintenance costs is the utilization of software metrics during the development phase, to help identify potential problem areas. Many new metrics have been proposed for OO systems, but only a few of them have been validated. The purpose of this research is to empirically explore the validation of three existing OO design cognitive complexity metrics, to assess their ability to predict maintenance time and a propose maintenance effort prediction model. This research reports the results of validating three metrics, Attributed Weighed Class Complexity (AWCC), Cognitive Weighted Response For a Class (CWRFC) and Cognitive Weighted Coupling Between Object (CWCBO). A controlled experiment was conducted to investigate the effect of design complexity (as measured by the above metrics) on maintenance time. Each of the three metrics was found to be useful in the experiment in predicting maintenance performance.***

Keywords—Maintenance effort prediction model, cognitive complexity metrics, Object-oriented metrics, software maintenance, metrics validation, predicting software maintenance time.

1. Introduction

The Object-Oriented (OO) technology is becoming an increasingly popular software development environment in recent years. This leads to more and more organizations introducing object-oriented methods and languages into their software development practices. The primary advantage of OOP (object-oriented programming) includes easier maintenance through better data encapsulation [7]. Although maintenance may turn out to be easier for programs written in OO languages, it does not mean that the maintenance burden will completely disappear [32]. Maintainability of software will continue to remain a critical area even in the object oriented era. The control of software maintenance costs can be approached in several ways. One approach is the use of software metrics during the development phase. Software complexity metrics can be used as indicators of the system quality and can help to identify potential problem areas [13] [23] [27]. Several metrics which are applicable during the design phase have been developed. Several studies have been conducted to examine the relationships between design complexity metrics and maintenance performance. Those studies concluded that design based complexity metrics can be used as predictors of maintenance performance. Chidamber and Kemerer (CK) [11] argue that, because of the fundamental difference between traditional approach and Object Oriented approach, software metrics developed with traditional methods in mind do not consider the OO features such as classes, inheritance, encapsulation, and message passing. Therefore, such metrics do not support key OO concepts, it is suitable to have new metrics especially designed to measure the unique aspects of the OO design [24].

Wang & Shao [30] argue that even the object oriented metrics can not reflect the real complexity of the OO code because they do not consider the cognitive characteristics in calculating code complexity of object oriented design. The cognitive complexity is defined as the mental burden on the user who deals with the code, like the developer, tester, maintenance staff etc. The cognitive complexity can be measured in terms of cognitive weights. Cognitive weights are defined as the extent of difficulty or relative time and effort required for comprehending given software, and measure the complexity of logical structure of software. A higher weight indicates a higher level of effort required to understand the software. A high cognitive complexity is undesirable due to several reasons, such as increased fault-proneness and reduced maintainability. Additionally, the cognitive complexity also provides valuable information for the design of OO systems. High cognitive complexity indicates poor design, which sometimes can be unmanageable [8]. In such cases, maintenance effort increases drastically. Basili et al. said that "metrics which reflect the specificities of the OO paradigm must be defined and validated" [6]. So, there is a need for a few empirical studies to investigate the relationship between the proposed metrics and OO design quality attributes such as maintainability [19]. Most of the earlier studies have investigated two metrics sets such as CK metrics suite [11] and the MOOD metrics [1] [2].

Li et al. [18] study the metrics proposed by CK [9] with reference to the maintenance effort in two commercial systems. They concluded that these metrics can be used as predictors of maintenance effort [19] [20]. Another experimental study [6] conducted to validate CK metrics as predictors of reliability (fault-proneness) found that five of the six metrics seem to be useful to predict class reliability during the early phases of the life cycle. Kolewe [18]

© 2013, IJARCSSE All Rights Reserved

confirms (based on a field study) that two of the metrics (class coupling and response for class) correlate with the defect densities. However, the work of CK is not without criticisms. Several researchers have pointed out ambiguities associated with some of these metrics [12] [17] [19].

OO designs are relatively richer in information. Therefore, metrics, if properly defined, can take advantage of that information available at any early stage in the life cycle. Unfortunately, most of the prior researcher does not exploit this additional information.

In order to overcome above said drawbacks of OO design metrics, Arockiam et al. [3] [4] [5] have proposed a cognitive complexity metric suite. This suite consist of three metrics, Attributed Weighed Class Complexity (AWCC) [3] Cognitive Weighted Response For a Class (CWRFC) [4] and Cognitive Weighted Coupling Between Object (CWCBO) [5], which are the focus of the current research. None of the studies have validated the proposed metrics empirically. The metrics have, however, been subjectively validated, where the metrics values are compared to existing metrics. The objective of the current research is to present the results of a study that assessed the validity of predicting maintenance time from the design complexity of a system as measured by the three metrics mentioned above. These metrics have also been analytically validated using the relevant mathematical properties specified by Weyuker [31]. This research, however, focuses only on the empirical study. The rest of this paper proceeds as follows: Section 2 discusses the design of the study, describing the dependent and independent variables, and the metrics that are to be validated. It presents the hypotheses to be tested, describes the subjects who participated in the study, and finally explains the data collection procedures, measurement instruments, and data collected. Section 3 presents an analysis of the data. Section 4 draws conclusions and makes suggestions for further work.

2. Design of the empirical study

The research design of this study, which suggests that design complexity, maintenance task, and programmer ability influence maintenance performance. Maintenance performance is the dependent variable where as design complexity and maintenance tasks are independent variables. The Figure 1 suggests that there may be some sorts of causal relationships on maintenance performance by design complexity and maintenance task. However, this study simply looked at to find existence of relationship to predict maintenance time. This study is also used to design a model to predict the maintenance effort. The empirical study was carried out using a controlled experiment in which students at Bharthidasan University (BDU) an Indian university participated as subjects.

2.1 Dependent variable

Maintainability is defined as the ease with which systems can be understood and modified [16]. In past studies, it has been measured as "number of lines of code changed [19] [20], time (required to make changes) and accuracy [14] [16], and "time to understand, develop, and implement modification" [26]. In this study, maintainability was measured as "time to understand, develop, and actually make modifications to existing programs [26]." We did not include accuracy in the maintenance measurement because of the following reasons: 1) an inverse relationship exists between time (for making changes) and accuracy. 2) To counter the criticism that the students used as participants lack motivation. The experiment was designed as a required assignment for a course. It was graded and the grade counted towards a component mark.

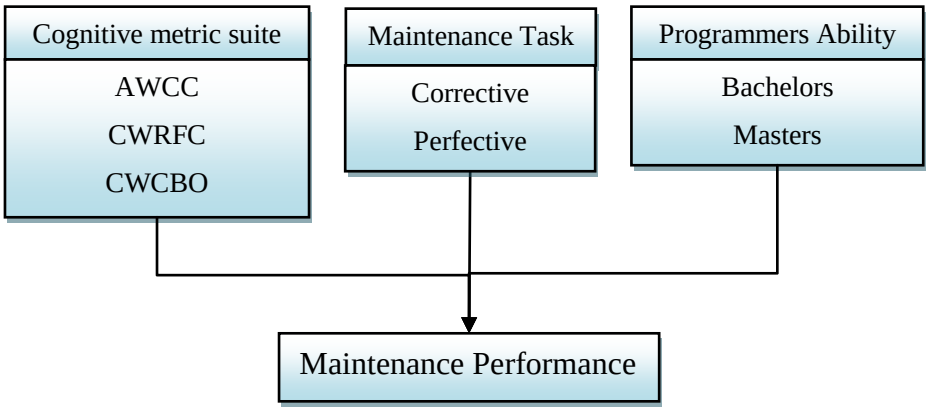

Figure 1: Research design**.**

2.2 Independent variables

2.2.1 Design complexity

Attribute Weighted Class Complexity (AWCC) [3] Cognitive Weighted Response for a Class (CWRFC) [4], and Cognitive Weighted Coupling Between Object (CWCBO) [5]. They were the measures of design complexity in this study. All these metrics have been subjectively validated by comparing their values with similar metrics and have been found to be a better metric. The three metrics are described below.

2.2.1.1 Attribute Weighted Class Complexity (AWCC)

A new metric namely Attribute Weighted Class Complexity (AWCC) has been proposed by Arockiam et al. [3]. AWCC is used to calculate the complexity of the class using the method complexity, attribute complexity of the class, and the inherited members' complexity. In AWCC, the cognitive weights have to be assigned for the attributes which are

© 2013, IJARCSSE All Rights Reserved

derived from the effort needed to understand their data types. If there are n attributes, m methods in a class and the class is derived from m1 number of classes then, the AWCC of that class can be calculated using the Eq (1).

$$AWCC = \sum_{i=1}^{n} AC_i + \sum_{j=1}^{m} MC_j + \sum_{k=1}^{m_1} ICC_k \qquad \text{Eq (1)}$$

where

AC is the attribute complexity,
MC is the method complexity,
ICC is the inherited class complexity.

Attribute Complexity (AC) is used to calculate the complexity of the attribute in the class by using the Eq (2).

$$AC = (PDT * W_b) + (DDT * W_d) + (UDDT * W_u) \qquad \text{Eq (2)}$$

where

PDT is the number of Primary Data Type attributes,
DDT is the number of Derived Data Type attributes,
UDDT is the number of User Defined Data Type attributes,
W_b is the Cognitive Weights of the PDT attributes,
W_d is the Cognitive Weights of the DDT attributes,
W_u is the Cognitive Weights of the UDDT attributes,

The weighting factor of attribute is based on the classification of cognitive phenomenon as described by [27] is as follows

Attribute types	Weights
Sub-Conscious Cognitive Attribute (PDT)	1
Meta Cognitive Attribute (DDT)	2
Higher Cognitive Attribute (UDDT)	3

The Method Complexity (MC) is calculated by assigning the cognitive weights proposed by Wang et al. [30] has assigned cognitive weights 1, 2, 3, and 2 to the sequence, branch, iteration and call structures respectively. J.Charles et al. [10] has also validated the weights proposed by Wang. Inherited Class Complexity (ICC) can be calculated using the Eq (3)

$$ICC = (DIT * C_L) * \sum_{k=1}^{s} RMC_k + RN_a \qquad \text{Eq (3)}$$

where,

S is the number of inherited methods,
RNa is the total number of Reused attributes,
RMC is the Reused Method Complexity,
DIT is the Depth of Inheritance Tree,
C_L is the Cognitive Load of L^{th} level.

C_L is the cognitive Load of L^{th} level which will differ from person to person according to the cognitive maturity level [25]. Here, the value of C_L is assumed to be 1 for simplicity

2.2.1.2 Cognitive Weighted Response For A Class (CWRFC)

A metric namely Cognitive Weighted Response For a Class (CWRFC) proposed by Aloysius et al. In CWRFC, the cognitive weights are assigned to the function call statement based on the effort needed to understand their type of function calls due to message passed by an object of that class. CWRFC is used to calculate the complexity of the class using the Response Set complexity. If there are m numbers of response sets in a class, then the CWRFC of that class can be calculated by using the Eq (4).

$$CWRFC = \sum_{j=1}^{m} RSC_j \qquad \text{Eq (4)}$$

where,

RSC is the response set complexity, which can be calculated by adding the set of all methods (M) in a class and set of methods (R) called by any of those methods as given in Eq (5).

$$RSC = M + \forall_i R_i \qquad \text{Eq (5)}$$

Based on the message passing, the Methods are divided into two categories Method with Argument (MWA) and Method without Argument (MOA). MOA is also known as Default Function (DF). Commonly in MWA, the arguments are passed in two ways namely, Pass By Value (PBV) and Pass By Reference (PBR). So RSC can be calculated by using the Eq (6).

$$R = DF * (CW_f + WF_d) + PBV * (CW_f + WF_v) + PBR * (CW_f + WF_r) \qquad \text{Eq (6)}$$

where,

DF is the total number of Default Function Call Statements,
PBV is the total number of Pass By Value Function Call Statements,
PBR is the total number of Pass By Reference Function Call Statements,
CW_f is the Cognitive Weights of the Function Call Statements,

© 2013, IJARCSSE All Rights Reserved

WF_d is the Weighting Factor of the DFC statements,
WF_v is the Weighting Factor of the PBV statements,
WF_r is the Weighting Factor of the PBR statements.

The cognitive weight of the Function Call (FC) Statement holds the value as 2 by [30].The weighting factor of different type of the FC statement is based on the classification of cognitive phenomenon as described by [28], is as follow

Function call Types	Weights
Sub- Conscious Cognitive Function Call Statement (DFC)	1
Meta Cognitive Function Call Statement (PBV)	2
Higher Cognitive Function Call Statement (PBR)	3

2.2.1.3 Cognitive Weighted Coupling Between Object (CWCBO)

CWCBO is an extension of the Coupling Between Object (CBO) metric proposed by CK. "Unnecessary object coupling needlessly decreases the reusability of the coupled objects", "Unnecessary object coupling also increases the chances of system corruption when changes are made to one or more of the coupled objects". The exiting CBO metric uses the count of number of objects the current classes coupled. Each couple is assigned a weight 1. This metric does not considered the various typed coupling [15]. Such as Control Coupling (CC), Global Data Coupling (GDC), Internal Data Coupling(IDC), Data Coupling(DC) and Lexical Content Coupling(LCC).

CWCBO can be calculated by using the Eq (7) as follows,

$$\mathrm{CWCBO} = (\mathrm{CC} * \mathrm{WFCC}) + (\mathrm{GDC} * \mathrm{WFGDC}) + (\mathrm{IDC} * \mathrm{WFIDC}) + (\mathrm{DC} * \mathrm{WFDC}) + (\mathrm{LCC} * \mathrm{WFLCC}) \quad \mathrm{Eq}(7)$$

Where

CC is the total number of modules that contain Control Coupling
GDC is the total number of modules that contain Global Data Coupling
IDC is the total number of modules that contain Internal Data Coupling
DC is the total number of modules that contain Data Coupling
LCC is the total number of modules that contain Lexical Content Coupling

The Weighting Factor of each type of coupling is calibrated using the method discuss in the previous section and the values are given as follows,

Types of Coupling	Cognitive Weight
WFCC	1
WFGDC	1
WFIDC	2
WFDC	3
WFLCC	4

2.2.2 Maintenance task

The second independent variable in the study is the maintenance task. Most of the researchers classify maintenance activities as adaptive, corrective, and perfective [21]. The need for adaptive maintenance arises when there are changes in hardware, operating systems, files, or compiler, which impact the system. Corrective maintenance is error-driven. This activity is equivalent to debugging, but it occurs after the system is placed in operation. Since, programs are never truly error free, Corrective maintenance is required throughout the life of a system. Perfective maintenance is user driven. Most perfective maintenance occurs in the form of report modifications to meet changing user requirements [21]. The bulk of maintenance activities are of this latter type. To be representative, two maintenance tasks were used in the study, one of which was perfective and the other was corrective.

2.2.3 Summary of Research Variables

Based on the above research model as shown in figure1, the main research objective was to focus on the relationship between the cognitive complexity metrics and the maintenance performance. Since, the design complexity is measured using cognitive complexity metrics, these metrics are to be validated and to find if these metrics are indeed valid metrics of design complexity in the contexts of both perfective and corrective maintenance tasks. This study has been conducted to find the relationship between design complexity and maintenance time and a to propose a model to predict the maintenance effort.

2.3 Hypothesis

The hypotheses for the study are derived from the following propositions:

P1. There is a relationship between the complexity of a system's design and the maintenance time.

© 2013, IJARCSSE All Rights Reserved

Propositions are generic statements made based on the research model discussed earlier (Figure1). P1 is a generic statement made based on the research model.

There are numerous ways to assess whether "a relationship" exists between two variables: t-test/ANOVA, correlation, regression, etc. For each of the metrics of interest in this study, three types of tests ANOVA, correlation, and regression were performed to assess whether a relationship exist and to see whether each complexity metric can be used as a reliable indicator of expected maintenance time. Each test is expressed in terms of a hypothesis. Both the Null (H_0) and the Alternate hypothesis (H_A) are shown.

The following hypotheses formalize these tests:

$H1_0$: There is no difference in the maintenance time required to make changes to systems, irrespective of whether they have low or high-complexity designs: $\mu 1 = \mu 2$.

$H1_A$: There is a difference in the maintenance time required to make changes to systems, depending on whether they have low or high-complexity designs: $\mu 1 ! = \mu 2$.

$H2_0$: There is no correlation between the complexity of a systems design and the maintenance time required to make changes to that system: $\rho = 0$.

$H2_A$: There is a nonzero correlation between the complexity of a systems design and the maintenance time required to make changes to that system: $\rho ! = 0$.

$H3_0$: There is no linear regression relationship between the complexity of a systems design and the maintenance time required to make changes to that system: $\beta_i = 0$.

$H3_A$: There is a nonzero linear regression relationship between the complexity of a systems design and the maintenance time required to make changes to that system: $\beta_i != 0$

We measured a systems complexity with each of the three metrics, AWCC, CWRFC, and CWCBO, and applied each of the hypotheses to each of the three metrics.

Proposition P1 and the resulting tests served as the main objective of our research, which was to validate the metrics (AWCC, CWRFC and CWCBO).

2.4 Study participants and the Experimental Treatments

The experiment was conducted over the duration of one semester and subjects came from a total of two sections (one section from bachelor's degree and the other sections from master's degree). The subjects participating in this research consisted of students taking the bachelor / master degree. The prerequisite of this course is that the students must have successfully studied OO programming language subject (JAVA). There were 122 students who had two years of programming language experience within which they had at least six months of OO experience. The students had the mean of 43.73 months of programming experience with stand deviation 17.3 and had a mean 11.9 months of object oriented experience with standard deviation 5.66 as shown in Table-1. This information was collected using a questionnaire that was filled out by all students.

Table 1: Summary of Subjects

	Mean	S
Yrs Exe Tot	43.73	17.3
Yrs Exe OO	11.98	5.66
N=122 Subjects		

Two independent treatments were used in the experiment, one involving perfective maintenance and the other involving corrective maintenance. Two versions of each treatment were constructed and designated as the "low-complexity" version and the "high-complexity" version based on their corresponding metric (AWCC, CWRFC and CWCBO) values. All the subjects from each of the two sections were assigned to work either on the low complexity or the high complexity version of each of the two treatments. Electronic and hard copies of the source code, along with the design specifications and proper documentation were given to the participants. The maintenance timings were self-reported and monitored.

Table 2: Allocation of Subjects to Treatments

	Perfective	Corrective	Total
Low Complexity	55	69	124
High Complexity	67	53	120
Total	122	122	244

The assignment of two treatments was based primarily on the desire to let a group have a low version of one treatment and the high version of the other treatment. The allocation of subjects to treatments is summarized in the Table 2. Fifty five students completed the low-complexity version of Treatment 1 (Perfective, "Quadrilateral"); Sixty seven completed the high-complexity version. Sixty nine students completed the low-complexity version of Treatment 2 (Corrective, "Tractor -Trailer"); Fifty three completed the high-complexity version. Nine students did not finish the experiment or didn't complete the profile survey.

© 2013, IJARCSSE All Rights Reserved

The first treatment involved a system called "Quadrilateral" (refer to Appendices A and B). The subjects were required to perform a perfective maintenance task on this system. This task involved adding new functionality to the system—computing the area and perimeter of the Quadrilateral. The second treatment involved a system called "Tractor - Trailer" (refer to Appendices C and D). The subjects were required to perform a corrective maintenance task on this system. (This task involved changing existing functionality of the system—changing the way taxes are computed for the Tractor -Trailer). The characteristics of these two systems as well as the corresponding metric values are summarized in the Table 3.

Table 3: Characteristics of the Two Programs

	Quadrilateral (perfective)		Tractor – Trailer (Corrective)	
	Low Complexity	High Complexity	Low Complexity	High Complexity
#Classes	2	1	1	3
#Methods	5	18	10	14
AWCC	10	26	31	46
CWRFC	5	18	10	14
CWCBO	13	16	12	20

All of the four system designs (two versions for each of the two systems) were pilot tested before the experiment. The pilot test was conducted with students of MPhil program in computer science at the same location where the experiment was performed

3. Data Analysis

The main objective in this experimental study was to empirically explore the validation of the three cognitive complexity metrics by assessing their ability to predict maintenance time. ANOVAs tests were conducted to determine if the mean maintenance times for the high- and low-complexity versions (categorized as high or low based on all three metric values) were significantly different. So the null hypotheses $H1_0$ for all three metrics are rejected. Correlation analysis and both simple as well as multiple regression analyses to examine the importance of each metric (AWCC, CWRFC and CWCBO) in determining maintenance time were conducted. The results in all cases were found significant and, thus, were able to reject the null hypotheses $H2_0$ and $H3_0$ for all three metrics. In order to validate the analyses data is divided into model building and holdout data sets [22]. Comparing the results obtained from the model-building data sets allowed us to gain confidence that our models and conclusions were valid.

3.1 Experimental Results

The following analyses were conducted on the gathered data.

3.1.1 Complexity versus Maintenance Time—Analysis of Variance (ANOVA)

Each subject received two treatments, a quadrilateral system requiring perfective maintenance (treatment 1) and a tractor-trailer system requiring Corrective maintenance (treatment 2). For each treatment, a single factor ANOVA was performed to verify. if the dependent variable (maintenance time) for the two groups (high complexity and low complexity systems) were equal.

Table 4: ANOVAs for Differences by Level of Complexity

Treatment	Mean Maintenance Time		F	P
	Low Complexity	High Complexity		
1	109.09	126.13	36.15	<0.00001
2	106.10	159.77	55.36	<0.00001

Table 5: ANOVAs for Differences by Scholar

Treatment	Mean Maintenance Time		F	P
	Low Complexity	High Complexity		
Bachelor	115.38	128.23	23.17	<0.00001
	126.53	215.41	1438.72	<0.00001
Master	103.45	123.97	18.77	<0.00001
	83.82	114.72	53.89	<0.00001

Table 4 shows the mean maintenance times for the first treatment quadrilateral system (perfective maintenance). The high-complexity version of the system had a higher mean time (about 126.13 minutes) compared to the low-complexity version (about 109.09 minutes). ANOVA was performed to test the statistical significance of this difference, and the results are tabulated. From the analysis we see that the P-value is less than 0.00001. Assuming as a null hypothesis that the complexity has no effect on the maintenance time, the probability of accepting null hypotheses is less than 0.00001. Therefore, as expected, this confirms proposition P1. Since the system was categorized as high or low complexity, based

© 2013, IJARCSSE All Rights Reserved

on the values of the three metrics, it is concluded that a system with greater AWCC (or CWRFC, or CWCBO) requires more time to perform a given maintenance task than the time required by a system with lower. Therefore, it is concluded that AWCC, CWRFC, and CWCBO are valid complexity metrics, and can reject the null hypotheses $H1_0$ for the first treatment for all three metrics. Similar analysis was done on the maintenance times for the second treatment— Tractor-Trailer (corrective maintenance). Table 5 shows the mean maintenance times for the second treatment. The high complexity version of the system had a higher mean time about 159.77 minutes compared to 106.10 minutes for the low complexity version. ANOVA was performed to test the statistical significance of this difference, and the results are shown in the table 4. From the analysis, the P-value is less than 0.00001. Therefore, it is concluded that AWCC, CWRFC, and CWCBO are valid complexity metrics and the null hypothesis $H1_0$ is rejected for the second treatment also for all three metrics.

It can be noted that, for both the treatments (perfective and corrective maintenance tasks), the null hypotheses $H1_0$ is rejected for all three metrics. Thus, metrics AWCC, CWRFC, and CWCBO can be used to predict system which needs a higher maintenance time. This is consistent with the requirement for a valid complexity metric. Thus it can be argued that the metrics AWCC, CWRFC, and CWCBO are valid OO Cognitive complexity metrics.

3.1.2 Impact of Cognitive Complexity Metrics— Correlation Analysis

A correlation analysis was applied in assessing the relationship between the metrics and maintenance time and the results are tabulated in Table 6. From the table 6 it is observe that in all three metrics high positive correlation except CWRFC with 60% of data. These correlations were calculated based on the model building data. Thus in almost every case the null hypothesis $H2_0$ is rejected for all three metrics and can conclude that the AWCC, CWRFC, and CWCBO are all useful empirical predictors of maintenance time.

Table 6: Correlation Analysis for Various Models – Building Sample Sizes

Model Building	AWCC	CWRFC	CWCBO
100%	1	1	1
60%	0.81247	0.481461	0.96845

3.1.2 Impact of Cognitive Complexity Metrics— Regression Analysis

In this section, the results of regression analysis conducted to investigate the importance of each of the three complexity metrics (independent variables) in determining the maintenance time (dependent variable) are discussed. Linear regression with one independent variable was performed for each of the three variables. Each of the variables AWCC, CWRFC, and CWCBO was found to have a statistically significant positive relationship with maintenance time. Table 6 summarizes the test statistics.

Table 7: Regression analysis – metrics and maintenance time

Variable	Test statistic n=244	p-value α=.05	β_i	Adjusted R^2
AWCC	11.43	<0.00001	1.69	0.35
CWRFC	5.95	<0.00001	2.69	0.12
CWCBO	10.74	<0.00001	6.65	0.32

Based on the results, the null hypothesis $H3_0$ is rejected for each of the metrics and can again conclude that all the three are valid predictors of maintenance time. The regression coefficient for the variable, CWRFC became negative. It was positive in the simple regression analysis, and was also expected to be positive from theoretical consideration. All of these are classic symptoms of multicollinearity. This problem appeared when the three variables AWCC, CWRFC, and CWCBO were used together in the multiple regressions.

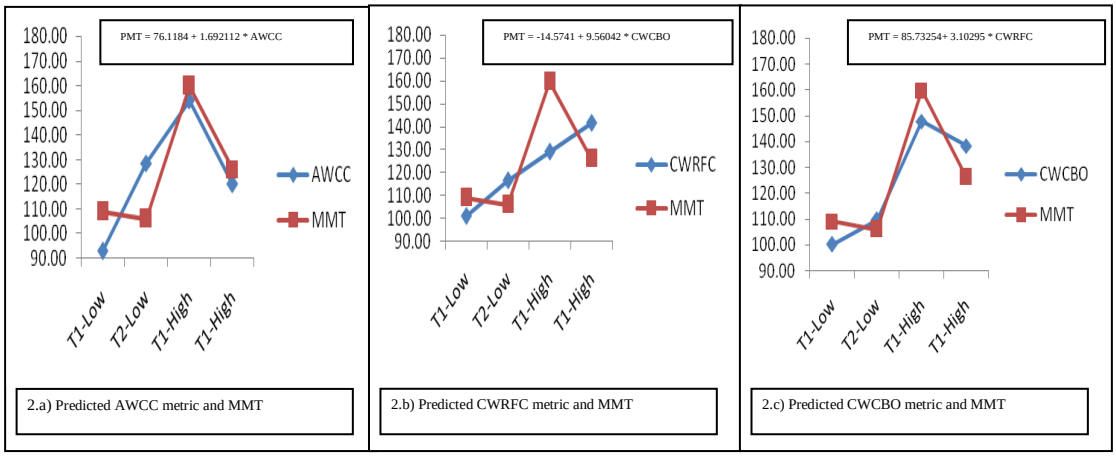

Figure 2. Comparisons of predicted and Mean Maintenance Time (MMT)

© 2013, IJARCSSE All Rights Reserved

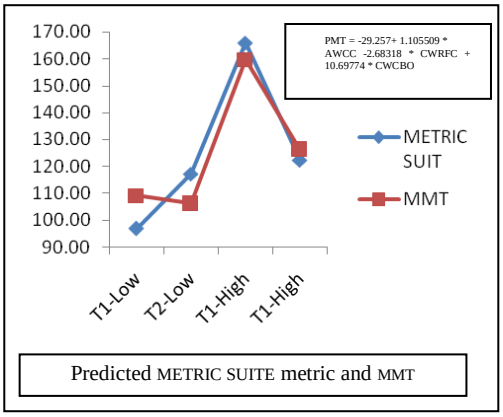

Figure 3. Predicted Maintenance Time (PMT)

From Table 7 it could be seen that AWCC and CWCBO explained more of the variance than CWRFC. The Table 7 and Figure 2 reflects result of Regression analysis between cognitive complexity metrics and maintenance time with variance 35% for AWCC (Predicted Maintenance Time(PMT = 76.1184 + 1.692112 * AWCC), 32% for CWCBO (PMT = -14.5741 + 9.56042 * CWCBO) and 12% for CWRFC (PMT = 85.73254+ 3.10295 * CWRFC).

Table 8: Regression analysis between cognitive complexity metric suite and maintenance time

Variable	Test statistic n=244	p-value α=.05	β_i	Adjusted R^2
AWCC	5.20	<0.00001	1.11	0.39
CWRFC	-2.83	0.004	-2.68	
CWCBO	4.07	<0.00001	10.70	

Multiple regression analysis with all three variables together was then performed to determine the combined explanatory power of these variables. The regression analysis as shown in Table 8 and figure 3 Confirms variance percentage of 39, which is greater than the percentage of variance of any other single metric value. This regression model is given in the following Eq 8,

PMT = -29.257+ 1.105509 *AWCC -2.68318 * CWRFC + 10.69774 * CWCBO Eq (8)

Thus, it is concluded that it would be more appropriate to use combined cognitive metric suite to predict maintenance effort.

3.2 Summary of Empirical Validation

To summarize, our analysis showed the metrics Attribute Weighted Class Complexity, Cognitive Weighted Response For Class, and Cognitive Weighted Coupling Between Object are empirically valid metrics for OO design complexity. Theoretically, systems with higher complexity need more time than systems with lower complexity in order to perform maintenance tasks. In this study, we categorized the relative complexity of the systems as high or low based on the values of the three metrics (AWCC, CWRFC, and CWCBO). Based on this classification, it's found that the systems that were categorized as high complexity needed a higher time (for maintenance) than those categorized as low complexity. Thus, it is concluded that the metrics AWCC, CWRFC, and CWCBO are useful and valid to measure the complexity of system design. The results are confident because three different measures (ANOVA, correlation, and regression) are used in various model-building in all cases. The obtained results also support the same conclusion. All three metrics are valid predictors of maintenance time. The cognitive complexity metrics AWCC, CWRFC, CWCBO were found to be a useful predictor of maintenance time.

4. Conclusions and Discussion

The main objective of this research was to empirically explore the validation of three object-oriented cognitive complexity metrics: Attribute Weighted Class Complexity (AWCC), Cognitive Weighted Response For Class (CWRFC), and Cognitive Weighted Coupling Between Object (CWCBO). For empirical validation, a controlled laboratory experiment was conducted to achieve the research objective. Analysis of variance (ANOVA), correlation, and single and multiple regression analysis were used to quantitatively analyze the experimental data. A summary of the major findings from the experiment is presented here. 1) Each of the three cognitive complexity metrics by themselves were found to be useful in measuring the design complexity. 2) The combined cognitive metric suite of AWCC, CWRFC and CWCBO found to be useful in measuring the design complexity. From the multiple regression models it is concluded that combined metric suite can be used to predict maintenance effort better than the individual metrics.

4.1 Limitations

The use of student as subjects is a question of concern. However, it is believed that, even with these issues, the study can provide useful information to software engineering practitioners and researchers. Because software maintenance is

© 2013, IJARCSSE All Rights Reserved

such a complex task, there are likely many issues that come into play besides the complexity of the design as measured by these metrics. The fact that they were found to be statistically significant predictors indicates that the study was successful within its scope.

4.2 Further Research

The experimental study can be extended as follows:

1. For the cognitive complexity metrics, a study can be conducted to separately capture the time required to understand the system and task, make changes, and test the changes. Also, an analysis of the different ways the changes are made can be performed. This can provide additional information on the impact of design complexity on detailed maintenance activities.
2. A longitudinal investigation of one or more actively maintained systems can be conducted. The cognitive complexity metrics being studied should be applied to the systems at the outset of the study and recomputed after each modification. Data can be gathered to evaluate how design complexity contributes to system deterioration, frequency of maintenance changes, system reliability, etc. This should provide useful information both to project managers as well as to system developers.
3. A study can be conducted to compare the three cognitive complexity metrics studied here to the other design metrics that have been proposed in the literature for their ability to predict maintenance performance.

References

[1] Abreu. B.F and Carapuca. R, "Candidate Metrics for Object- Oriented Software with in a Taxonomy Framework", J. System Software, 1994, vol. 26, pp. 87-96.

[2] Abreu. B.F, Goulao. M., and Esteves. R, "Toward the Design Quality Evaluation of Object-Oriented Software Systems," Proceedings Fifth International Conference Software Quality, 1995.

[3] Arockiam. L, Aloysius. A, "Attribute Weighted Class Complexity: A New Metric For Measuring Cognitive Complexity Of OO Systems", Proceedings of International Conference on Computational Intelligence and Cognitive Informatics, Indonesia, October 2011.

[4] Aloysius .A, Arockiam .L, "Cognitive Weighted Response For a Class: A New Metric for Measuring Cognitive Complexity of OO Systems", 11th IEEE International Conference on Cognitive Informatics & Cognitive Computing in Kyoto, Japan during August 22-24, 2012. (Accepted)

[5] Aloysius. A, Arockiam. L, "Cognitive Weighted Coupling Between Objects", International Journal of Software Engineering and Technology, May, 2012, Print: ISSN 0974 – 9748 & Online: ISSN 0974 – 9632. (Communicated)

[6] Basili. V.R, Briand. L.C, and Melo. W.L, "A Validation of Object Oriented Design Metrics as Quality Indicators," IEEE Transactions Software Engineering, 1996, pp. 751-761.

[7] Booch. G, "Object –Oriented Development," IEEE Transactions on Software Engineering, 1986.

[8] Briand L C, Bunse C, Daly J W 2001 A controlled experiment for evaluating quality guidelines on maintainability of object oriented design. IEEE Transactions on Software Engineering, 2(6): 513–530.

[9] Briand. L.C, Wust. J, Daly. J.W, and Porter. D.V, "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems," The J. System and Software, 2000, vol. 51, pp. 245-273.

[10] Charles Selvaraj. J, Aloysius. A, and Arockiam. L , "A Comparison of Proposed Cognitive weights for control structures and object oriented programming languages", Proceedings of International Conference on Advanced Computing ICAC09, 2009, pp.380-385.

[11] Chidamber, S.R and Kemerer. C.F, "A Metrics Suite for Object Oriented Design," "IEEE Transactions Software Engineering,1994, pp. 476-493.

[12] Churcher. N.I., Shepperd. M.J, "Comments on "A Metrics Suite for Object Oriented Design," IEEE Transactions Software Engineering, 1995, vol. 21, pp. 263-265.

[13] Coleman. D, Lowther. D, and Oman. P, "The Application of Software Maintainability Models in Industrial Software Systems", Journal of System Software, 1995, vol. 29, pp. 3-16.

[14] Crutis. B, Shepperd. S.B, Milliman. P, Borst. M.A, and Love. T, "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics," IEEE Transactions Software Engineering, 1979, pp. 96-104.

[15] Edward Berard. V "Essays on object-oriented software engineering (vol. 1)" Berard Software Engineering, Prentice-Hall, 1993, ISBN:0-13-288895-5.

[16] Gibson. V.R and Senn. J.A, "System Structure and Software Maintenance Performance," Comm. ACM, 1989, pp. 347-358.

[17] Hitz. M and Montazeri. B, "Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective," IEEE Trans. Eng., 1996, vol. 22, no.4, pp. 267-271.

[18] Kolewe. R, "Metrics in Objects-Oriented Design and Programming," Software Development, 1993, pp. 53-62.

[19] Li. Wand Henry. S, "Object Oriented Metrics that Predict Maintainability," J. System Software, 1993, pp. 111-122.

[20] Li. Wand Henry. S, Kafura. D and Schulman. R, "Measuring Object-Oriented Design," J. Object Oriented Programming, 1995, vol. 8, no.4, pp. 48-55.

[21] Lientz. B.P and Swanson. E.B, "Software Maintenance Management. Reading", Mass: Addison-Welsey, 1980.

[22] Neter. J, Wasserman. W., and Kutner. M, Applied Linear Statistical Models: Regression, Analysis of Variance,

© 2013, IJARCSSE All Rights Reserved

and Experimental Designs, third ed. Boston: Irwin, 1990.
[23] Pearse. T and Omen. P, "Maintainability Measurements on Industrial Source Code Maintenance Activities", Proc. Int'l Conf. Software Maintenance, 1995, pp.295-303.
[24] Rajendra K. Bandi, Vijay K. Vaishnavi, "Predicting Maintenance Performance Using Object-Oriented Design Complexity Metrics", IEEE Transactions on software engineering,vol.29, No.1, Jan 03.
[25] Ranjeeth. S, Ramu Naidoo "An Investigation into the Relationship Between the level of Cognitive Maturity And the types of errors made By Students In A Computer Programming" College Teaching Methods & Style Journal-Second Quarter,2007,pp,31.40
[26] Rising. L.S, "Information Hiding Metrics for Modular Programming Languages," PhD dissertation, Arizona State Univ., 1992.
[27] Sneed. H.M, "Applying Size Complexity and Quality Metrics to an Object-Oriented Application," Proc. European Software Control and Metrics conf. Software certification Programme in Europe (ESCOM-SCOPE), 1999.
[28] Wang. Y, "On Cognitive Informatics." IEEE International Conference on Cognitive Informatics, 2002, pp.69-74.
[29] Wang. Y, "On Cognitive Informatics." IEEE International Conference on Cognitive Informatics, 2002, pp. 69-74.
[30] Wang. Y and Shao. J, "A new measure of software complexity based on cognitive Weights." IEEE Canadian Journal of Electrical and Computer Engineering, 2003, pp.69-74.
[31] Weyuker. E.J., "Evaluating Software Complexity Measures," IEEE Trans. Eng., 1988, vol. 21, pp. 1357-1365
[32] Wilde. N and Huitt. R., "Maintenance Support for Object- Oriented Programs, "IEEE Transactions on Software Engineering, 1992, pp. 1038-1044, vol.12, no. 2, pp. 211-221

Authors

Dr. L. Arockiamis working as Associate Professor in the Department of Computer Science, St.Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu, India. He has 22 years of experience in teaching and 14 years of experience in research. He has published 107 research articles in the International / National Conferences and Journals. He has also presented 2 research articles in the Software Measurement European Forum in Rome and one in Indonesia. He has chaired many technical sessions and delivered invited talks in National and International Conferences. He has authored a book on "Success through Soft Skills". His research interests are: Software Measurement, Cognitive Aspects in Programming, Data Mining and Mobile Networks.

<A. Aloysius is working as Assitant Professor in Department of Computer Science, St.Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu, India. He has 12 years of experience in teaching and research. He has published 15 research articles in the National / International conferences and journals. He has acted as a chair person for many national and international conferences. He is currently pursuing doctor of philosophy programme and his current area of research is cognitive complexity metrics in software design.

© 2013, IJARCSSE All Rights Reserved