Models for Maintenance Effort Prediction with Object-Oriented Cognitive Complexity Metrics

T. Francis Thamburaj[1], Dr. A Aloysius[2]
[1, 2] Assistant Professor, Dept. of Computer Science,
St. Joseph's College, Tiruchchirappalli – 620 002, India
francisthamburaj@gmail.com, aloysius1972@gmail.com

Abstract- **Software maintenance is the most desired, but most elusive and difficult task in software engineering. The cost of maintenance is as high as 60% to 80% of the total cost of the software. So, plenty of researches are going on in software maintenance. Though, object-oriented paradigm has made it easier, it remains the critical hotspot of research. One way of grappling with the maintenance problem, is to use the complexity metrics. Many studies were made to understand the relationship among complexity metrics, cognition, and maintenance. This paper wrestles with four newly proposed object-oriented cognitive complexity metrics to develop maintenance effort prediction models through various statistical techniques. Empirical study designs are made with hypotheses and experimented. Discussion on results prove the maintenance effort prediction models are more robust, more accurate, and can be employed to estimate the maintenance effort.**

Keywords—maintenance effort prediction; cognitive complexity metrics; object-oriented metrics; software maintenance

I. Introduction

High software quality is the eternal hallmark of software engineering. Among the several software qualities maintainability and understandability are the most important and key qualities that are desired in the industry due to overall reduction in cost and effort [1]. The maintenance cost is as high as 60% to 80% of the total cost of the software [2]. In fact, it is the key for the survival of the product through the evolution as it faces many challenges from the constantly changing environments [3].

Software maintenance is the most desired, but most elusive and difficult task. Software maintenance is defined, as per ISO/IEC 9126 and IEEE 1219, "the process of modifying the software system or component after delivery to correct faults, improve performance or other attributes or adapt to a changed environment" [4]. There are four categories of maintenances, namely, corrective, perfective, adaptive, and preventive maintenance [3]. The corrective maintenance consumes about 21% and the adaptive and perfective maintenance takes about 75% of the maintenance effort. The perfective maintenance is the core problem of software maintenance during evolution [5].

Maintenance difficulty depends upon the complexity of the software system. To reduce the complexity, Object-Oriented (OO) paradigm is adopted which raises the cognition and eases the maintenance tasks [6]. Even the OO metrics, according to Wang and Shao [7], cannot reflect the real complexity of OO code since they consider only the structural aspect and do not bother about the cognitive aspects in calculating the code complexity. In fact, maintenance should be measured not only in terms structural complexity but also the amount of time taken to understand the program (cognitive aspect) and the effort needed to do the maintenance task [8].

In spite of all these measures, maintenance burden remains a critical area of research [9]. The relationship between the maintenance effort and OO metrics is complex and non-linear [10]. The cognitive weighted OO metrics further complicate it. Hence, the modelling and prediction of maintenance effort remain the hotspot of research and a lot of statistical models and sophisticated techniques are designed.

This paper explores the relationship between the maintenance and four newly proposed complexity metrics by the authors. The rest of the article is divided as follows: section 2 portrays the literature survey and section 3 describes the study design. Section 4 deals with the empirical experiment and section 5 discusses the results. The last section concludes with major findings and possible future works.

II. Survey of Literature

Several studies have been conducted to examine the relationships among design complexity, program cognition and maintenance. As early as in 1976, Swanson et al., have categorized maintenance into corrective, perfective, adaptive, and later with Lientz added preventive maintenance [11]–[12] . Li and Sallie validated OO metrics, Lanning et al applied canonical correlation analysis, Kemerer and later Polo et al., used empirical studies and found that the prediction of maintenance effort is possible with OO product metrics [13]–[16]. Benestad et al. studied, how class-level measures of structural properties can be used to assess the maintainability of a software product as a whole [17].

In 2012, Al-Fawareh, studied various OO techniques like polymorphism, inheritance, dynamic biding, complex dependencies etc., from maintenance perspective [18]. Aloysius et al., in 2013, utilized three cognitive complexity metrics to develop a maintenance effort prediction model [9]. Michura et al. identified valuable attributes in determining the difficulty in implementing changes during maintenance [19]. Ogheneovo, in 2014, studied different operating systems and found a strong correlation between software complexity and maintenance cost [2]. In 2015, Lee et al. identified 17 effort estimation factors for corrective maintenance by exploring the expert software maintenance estimators' knowledge [20]. In 2016 Palak Diwan predicted the maintenance of OO system using fuzzy logic [21].

III. Empirical Study Design

The empirical study design suggests that the design complexity, maintenance task, and the programmer ability influence the maintenance performance. The dependent variable is the maintenance performance and the independent variables are the design complexity and maintenance tasks. The aim of this empirical study is to look for the existence of some causal relationship between the dependent and independent variables, and consequently to develop a prediction model for maintenance effort. This research study was carried out using a controlled experiment in which the Bharathidasan University in India took part as subjects. The experiment was designed as a required assignment for a course. It was graded and the grade was counted towards a component mark.

The dependent variable, namely maintenance performance, can be measured in terms of number of lines of code changed; accuracy along with the time required to change; and time taken to understand, develop, and modify the existing programs [8]. This study did not include the accuracy factor due to the fact that there is an inverse relationship between time to make changes and accuracy. Further, it helps to counter the criticism that the participating students lack motivation.

The first independent variable, namely the type of maintenance tasks, the corrective and perfective maintenance, among the four types explained in the introduction section, were used in this study.

The second independent variable, namely the design complexity, consists of four object-oriented cognitive metrics. They are Cognitive Weighted Attribute Hiding Factor (CWAHF), Cognitive Weighted Method Hiding Factor (CWMHF), Cognitive Weighted Polymorphic Factor (CWPF), and Cognitive Weighted Cohesion (CWC). Francis Thamburaj and Aloysius, the authors of this article, proposed these four design metrics, mathematically defined, calibrated their cognitive weights, experimented with case studies, theoretically validated with Weyuker's 9 properties for valid software metric, Abreu's 7 criteria for robust object-oriented metrics, proved more accurate by comparative studies with similar metrics, and demonstrated the statistical soundness with correlation analysis. The mathematical formulations of these equations are as follows:

$$CWMHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_h(C_i) + \sum_{i=1}^{TC} M_v(C_i)} \qquad (1)$$

$$CWAHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_h(C_i) + \sum_{i=1}^{TC} A_v(C_i)} \qquad (2)$$

$$CWPF = \frac{\sum_{i=1}^{TC} CWM_o(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) * DC(C_i)] * ACW} \qquad (3)$$

$$CWCoh = \left(\frac{a}{kl}\right) CW_{degree} \qquad (4)$$

For the sake of the brevity, the details are not provided here, but it can be found in [22]–[25].

IV. Empirical Experiment

The objective of the experiment, the existence of relationship between design complexities and maintenance tasks, can be assessed in many ways like ANOVA, correlation, and regression tests. If there is a relationship, each of the four design complexities can be used as a reliable indicator of expected maintenance effort. The hypotheses for this empirical experiment are derived from the following propositions:
$H1_0$: There is no difference in maintenance time required to make changes to system with low or high complexity design: $\mu 1 = \mu 2$.
$H1_A$: There is difference in maintenance time required to make changes to system with low / high complexity design: $\mu 1 != \mu 2$.
$H2_0$: There is no correlation between system complexity and maintenance time required to make changes to system with low or high complexity design: $\rho = 0$.
$H2_A$: There is correlation between system complexity and maintenance time required to make changes to system with low or high complexity design: != 0.
$H3_0$: There is no linear regression relationship between system complexity and maintenance time required to make changes to system with low or high complexity design: $\beta_i = 0$.
$H3_A$: There is linear regression relationship between system complexity and maintenance time required to make changes to system with low or high complexity design: $\beta_i != 0$.

The experiment was conducted over one semester with two groups of students doing their bachelor's and master's degree. All of them have completed one semester course on Java. Based on their computer, especially Java programming experience and semester performance, 92 (43UG+49PG) students were chosen to participate in the experiment.

Two independent treatments, one for corrective and other for perfective, were used for the experiment. Based on the complexity metric CWAHF, CWMHF, CWPF, and CWC values, high and low complexity versions were constructed for each treatment. The following Table I provides the details of the participation.

TABLE I
Participant Distribution

Complexity	Corrective Maintenance	Perfective Maint.	Total
Low	49 (23UG+26PG)	51 (22UG+29PG)	100
High	43 (19UG+24PG)	41 (21UG+20PG)	84
Total	92 (42UG+50PG)	92 (43UG+49PG)	184

The first treatment, "Shape" system that calculates area, volume, etc., is used for perfective maintenance. The subjects were asked to add new routines for cylinder, sphere and cone. The second treatment, "Salary" system that calculates net salary is used for corrective maintenance. The subjects were asked to correct the method of salary calculation based on a set of rules. The characteristics of the treatments are given in Table. II

TABLE II
Characteristics of the Perfective and Corrective Treatments

	Perfective (Shape)		**Corrective (Salary)**	
	Low Complexity	***High Complexity***	***Low Complexity***	***High Complexity***
#Classes	4	4	4	4
#Methods	7	7	8	7

CWAHF	0.888	0.154	0.958	0.4
CWMHF	0.962	0.5	0.971	0.5
CWPF	0.6	0.166	0.514	0.2
CWC	0.714	0.125	0.8	0.141

V. Results and Discussion

For each treatment a single factor ANOVA was performed to check the equality of dependent variable (maintenance time) for high and low complexity groups. Table III shows the mean maintenance times in minutes for the high complexity versions (56.68, 70.42) are higher than the low complexity versions (36.80, 59.14) for both treatments and the P-value, showing the significance, is less than 0.00001. This confirms the $H1_A$ hypothesis, namely, there is maintenance time difference between low and high complexity designs for perfective as well as corrective treatments.

TABLE III
ANOVAs for Differences by Level of Complexity

Treatment	Mean Maintenance Time		F	P
	Low Complexity	*High Complexity*		
Perfective	36.8039	56.6829	131.206	< 0.00001
Corrective	59.1370	70.4242	140.918	< 0.00001

The correlation analysis was done to assess the relationship between the metrics and maintenance time. The results are tabulated in Table IV. All four metrics are negatively correlated or inversely related to maintenance time. That is, the maintenance time decreases as the complexity of the metrics increases. This is what is expected and intuitively truthful. When the system is more cohesive, attributes and methods are localized within the class, and polymorphism is high, both perfective and corrective type of modifications becomes easier and takes less time. The correlation results reject $H2_0$, the null hypothesis.

TABLE IV
Correlation Analysis for Various Models – Building Sample Sizes

Model Building	CWAHF	CWMHF	CWPF	CWC
100%	-0.495*	-0.645*	-0.676*	-0.602*
80%	-0.491*	-0.640*	-0.668*	-0.598*
60%	-0.504*	-0.656*	-0.682*	-0.614*

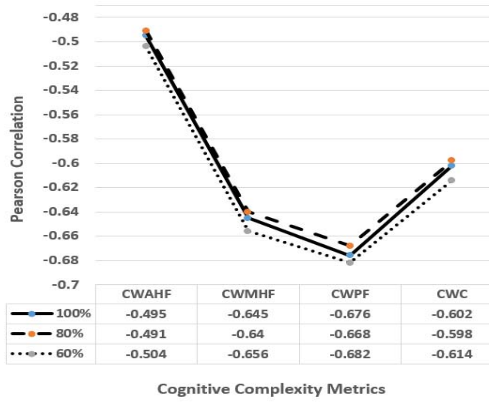

Fig. 1 Correlation Chart for Various Models

The Linear Regression analysis is also done for each of the four cognitive complexity metrics (independent variables) in determining the maintenance time (dependent variable) and found to have statistically significant negative relationship. Table V provides test statistics summary. Based on the results, the null hypothesis $H3_0$ is rejected for each of the metrics and concludes that all the four metrics are valid predictors of maintenance time.

TABLE V
Regression Analysis for Metrics and Maintenance Time

Variable	Test Statistic n = 184	p-value α = 0.05	β_i	Adjusted R^2
CWAHF	-7.676	< 0.00001	-0.495	0.240
CWMHF	-11.377	< 0.00001	-0.645	0.412
CWPF	-12.372	< 0.00001	-0.676	0.454
CWC	-10.171	< 0.00001	-0.602	0.359

Predictive Maintenance Time (PMT) with each cognitive complexity metric can be derived from the data provided in Table V. The predicting linear equations are listed below:

$$PMT = 71.351 – 0.495 * CWAHF \text{ with 24\% variance} \quad (5)$$
$$PMT = 91.038 – 0.645 * CWMHF \text{ with 41\% variance} \quad (6)$$
$$PMT = 78.894 – 0.676 * CWPF \text{ with 45\% variance.} \quad (7)$$
$$PMT = 70.877 – 0.602 * CWC \text{ with 36\% variance.} \quad (8)$$

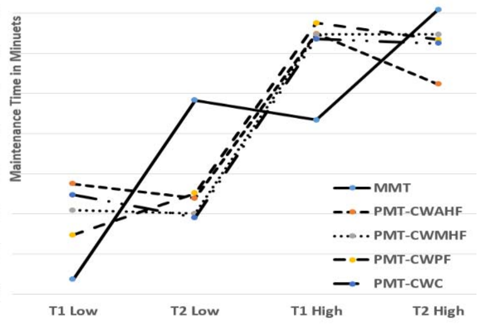

Fig. 2 Regression Analysis for Individual Metric

The multiple regression was also performed for all the four cognitive complexity metrics as a suite. Table VI provides the results. The CWMHF complexity metric was excluded by SPSS tool due to multicollinearity, implying that the combination of CWAHF, CWPF, and CWC yields more precise prediction than the combination of all four complexity metrics.

TABLE VI
Regression Analysis for CW Metric Suite and Maintenance Time

Variable	Test Statistic n = 184	p-value α = 0.05	β_i	Adjusted R^2
CWAHF	35.377	< 0.00001	2.158	0.726
CWMHF	Excluded			
CWPF	-12.072	< 0.00001	-1.788	
CWC	-5.052	< 0.00001	-0.978	

The corresponding PMT linear equation for the proposed cognitive weighted metric suite, with variance of 72.6%, is

$$PMT = 73.0 +111.181*CWAHF –161.132*CWPF –53.572*CWC \quad (9)$$

The variance percentage of the suite is much greater than any other single complexity metric value. Hence, it is concluded that it would be more appropriate to use combined cognitive complexity metric suite to predict maintenance effort.

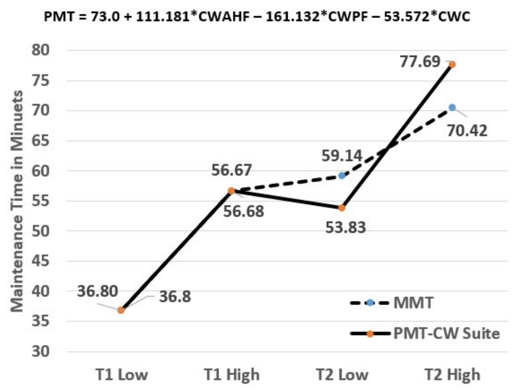

Fig. 3 Cognitive Metric Suite PMT and MMT

Using (9), the PMTs are calculated for all the four programs of the two treatments. Fig. 3, portrays the results graphically. The PMT for perfective maintenance (T1) is aligned more with the MMT than the corrective maintenance (T2). For T2, the PMT is lower than MMT for low complexity, while it is higher in case of high complexity. In spite of the variations, prediction accuracy is very high with deviations ranging from 5 to 8 minutes only.

VI. Conclusion

The main objective of this research, the empirical validation of CWAHF, CWMHF, CWPF, and CWC complexity metrics, through controlled experiments are achieved with great validity and high accuracy. Various statistical measures like ANOVA, correlation, single and multiple regression analyses were used to quantitatively analyze the experimental data. The major and important findings are that each of four proposed complexity metrics were useful in measuring the design complexity. Also, the Cognitive Weighted Suite of Object Oriented Paradigm (CWSOOP), is proved to be much better and more accurate predictor of both perfective and corrective maintenance effort than the individual complexity metric. The other achievement of this research experiment is the development of five different maintenance effort prediction models, of which four models use the individual complexity metric and the fifth uses combined metric values of CWSOOP.

For future work, the experiment can include other cognitive complexity metrics to bring out a bigger suite of maintenance effort estimation model. To confirm the results, programmers from software industry can be utilized. Further, large systems from open sources can be studied for the same purpose.

References

[1] Sommerville, I., "Software Engineering," 7th Ed., Addison Wesley, 2004.

[2] Ogheneovo, Edward E., "On the Relationship between Software Complexity and Maintenance Costs," Jr. of Computer and Communications, vol. 2, no. 14, pp. 1-16, 2014.

[3] Mamdouh Alenezi and Mohammad Zarour, "Does Software Structures Quality Improve over Software Evolution? Evidences from Open-Source Projects," Int. Jr. of Computer Science and Information Security (IJCSIS), vol. 14, pp. 61-75, February 2016.

[4] IQBBA, "Standard Glossary of Terms Used in Software Engineering," Int. Qualifications Board for Business Analysis, 2011.

[5] Bennett, Keith H., and Václav T. Rajlich, "Software Maintenance and Evolution: A Roadmap," In Proc. of the Conference on the Future of Software Engineering, ACM, pp. 73-87, 2000.

[6] Booch. G, "Object–Oriented Development," IEEE Transactions on Software Engineering, vol. 12, no. 2, pp. 211-221, 1986.

[7] Yingxu Wang and Jingqiu Shao. J, "A New Measure of Software Complexity Based on Cognitive Weights," IEEE Canadian Jr. of Electrical and Computer Engineering, pp.69-74, 2003.

[8] Aloysius A., "A Cognitive Complexity Metrics Suite for Object Oriented Design," PhD Thesis, Bharathidasan University, Trichy, India, 2012.

[9] A. Aloysius, L. Arockiam, "Maintenance Effort Prediction Model Using Cognitive Complexity Metrics," Int. Jr. of Advanced Research in Computer Science and Software Eng., vol. 3, no. 11, pp. 1599-1608, 2013.

[10] Kaur, Arvinder, Kamaldeep Kaur, and Ruchika Malhotra, "Soft Computing Approaches for Prediction of Software Maintenance Effort," Int. Jr. of Computer Applications, vol. 1, no. 16, 2010.

[11] Swanson, E. Burton, "The Dimensions of Maintenance," In Proceedings of the 2nd International Conference on Software Engineering, pp. 492-497, IEEE Computer Society Press, 1976.

[12] Lientz B. P., Swanson E. B., "Software Maintenance Management," Addison Wesley, Reading, MA, 1980.

[13] Li, Wei, and Sallie Henry, "Object-oriented Metrics that Predict Maintainability," Jr. of Systems and Software, vol. 23, no. 2, pp. 111-122, 1993.

[14] Lanning David L., and Khoshgoftaar, Taghi M., "Modeling the Relationship Between Source Code Complexity and Maintenance Difficulty," IEEE Computer, 1994.

[15] Chris F. Kemerer, "Software Complexity and Software Maintenance: A Survey of Empirical Research," Annals of Software Engineering, vol. 1, no. 1, pp. 1-22, 1995.

[16] Polo, Macario, Mario Piattini, and Francisco Ruiz, "Using Code Metrics to Predict Maintenance of Legacy Programs: A Case Study," In Proceedings of the IEEE Int. Conference on Software Maintenance (ICSM'01), IEEE Computer Society, pp. 202, 2001.

[17] Benestad, Hans Christian, Bente Anda, and Erik Arisholm, "Assessing Software Product Maintainability Based on Class-Level Structural Measures," Int. Conference on Product Focused Software Process Improvement, Springer Berlin Heidelberg, pp. 94-111, 2006.

[18] Al-Fawareh, Hamed J., "Modeling an Object Oriented for Maintenance Purposes," Int. Jr. of Computers and Technology, vol. 3, no. 3, pp. 401-405, 2012.

[19] John Michura, Miriam A. M. Capretz, and Shuying Wang, "Extension of OO Metrics Suite for SW Maintenance," Hindawi Pub. Corporation, ISRN Software Engineering, vol. 2013, 14 pages, 2013.

[20] Lee, Michael J., Marcus A. Rothenberger, and Ken Peffers, "Effort Estimation Factors for Corrective Software Maintenance Projects: A Qualitative Analysis of Estimation Criteria," Jr. of Information Technology Theory and Application, vol. 16, no. 2, 2015.

[21] [Diwan, 16] Palak Diwan, "Predicting Maintainability of Object-Oriented System using Fuzzy Logic," International Journal of Scientific and Research Publications, vol. 6, no. 3, March 2016.

[22] Francis Thamburaj and A. Aloysius, "Cognitive Weighted Method Hiding Factor Complexity Metric," Int. Jr. of Computer Science and Software Engineering (IJCSSE), ISSN (Online): 2409-4285, vol. 4, no. 10, pp. 272-279, October 2015.

[23] Francis Thamburaj and A. Aloysisus, "Cognitive Perspective of Attribute Hiding Factor Complexity Metric," Int. Jr. of Engineering and Computer Science, IF 3.093, GIF 5.64, ISSN: 2319-7242, vol. 4, no. 11, pp. 14973-14979, November 5th 2015.

[24] Francis Thamburaj and Aloysius, A., "Cognitive Weighted Polymorphism Factor: A Comprehension Augmented Complexity Metric," WASET Conf. in Kyoto, Int. Jr. of Computer, Electrical, Automation, Control and Information Engineering, PISSN:2010-376X, EISSN:2010-3778, vol. 9, no. 11, pp. 2199 – 2204, November 12th 2015.

[25] Francis Thamburaj and A. Aloysius, "Cognitive Weighted Cohesion Complexity Metric," Int. Jr. of Control Theory and Application (IJCTA), SCOPUS Indexed, July 24th 2016.