IJCTA, 9(27), 2016, pp. 575-583 ISSN: 0974-5572

On Validating Cognitive Weighted Attribute Hiding Factor Complexity Metric

Francis Thamburaj* and A. Aloysius*

ABSTRACT

Software complexity metric is widely accepted tool to control and assure software quality. Plethora of software metrics can be found in the software engineering literature. However, most of them are not properly validated. But the success of the software complexity metric truly lies in the validation process, which proves its correctness, truthfulness, accuracy, and robustness. This paper validates the proposed complexity metric called Cognitive Weighted Attribute Hiding Factor (CWAHF). Multiple validations such as the case studies to prove the applicability of the complexity metric, theoretical validations against Weyuker's nine properties, Abreu's seven criteria and Kitchenham benchmarks to prove the robustness of the metric. In order to corroborate the theoretical validation, the empirical validation is done. The comparative study is conducted to show the better accuracy. The correlation analysis is performed to show the statistical validity. All the validations proved that the complexity metric Cognitive Weighted Attribute Hiding Factor is truly a valid, more robust, more realistic, more accurate and more comprehensive complexity metric.

Keywords: Cognitive Weighted Attribute Hiding Factor, Software Metric Validation, Object-Oriented Software Complexity Metrics, Software Cognitive Complexity Metrics, Software Metrics.

1. INTRODUCTION

The reign of object-oriented (OO) paradigm is still in high esteem and attracts most of the developers, due to its inherent power to represent the real world in easy and elegant way, clearly structured program, easy comprehension, high reusability and extensibility due to inheritance, contextual processing due to polymorphic features. Above all these reasons, data safety through the attribute hiding, which is part of the information-hiding, plays the important role by serving as the foundation for several guidelines in software design in the OO programming world [1].

The success of the software product is highly dependent on the attribute hiding, which protects the data against any accidental or unwanted changes. The attribute hiding is achieved by encapsulating the instance variables that hold data values. Normally, 'class' block is used to encapsulate. But encapsulation and information hiding are not the same [2]. The instance variables may be encapsulated but may still be totally or partially visible to other classes and packages. Also, information can be hidden without being encapsulated, and vice-versa [3]. So, the attribute hiding is actually implemented by the combination of class encapsulation and scoping of the attributes.

Among the classical OO metric suites, MOOD metric suite by Abreu et al. brings out two different metrics for attribute hiding, namely, Attribute Hiding Factor (AHF) and Attribute Hiding Effectiveness Factor (AHEF). The AHF is defined as the ratio of the sum of the invisibilities of all attributes defined in all classes to the total number of attributes defined in the software system [4]. While AHF measures the general level of attribute hiding, the AHEF

* Department of Computer Science, St. Joseph's College, Bharathidasan University, Trichy, *Emails: francisthamburaj@gmail.com, aloysius1972@gmail.com*

measures how well the hiding succeeds. AHEF is the ratio of classes that access attributes, to the classes that can access attributes[5]. Henderson Seller has proposed the Number Of Attributes (NOA) as a metric in 1996 [6]. In the QMOOD metric suite by Bansiya et al. proposed the Data Access Modifier (DAM) metric. It is the ratio of private attributes to the total number of attributes declared in the class [7]. Pasupathy and Bhavani [8] have proposed many attribute related class oriented metrics in 2013. Vinay Singh et al. have defined the Measure Of Aggregation (MOA) as the count of attributes whose types are user-defined classes [9]. In 2014, Srinivasan et al. defined the metric called Attribute-Per-Class Factor (APCF) as the percentage of private attribute, excluding inherited attributes [10]. In 2015 Chhillar et al. [11] defined a set of member access control metrics consisting of Member Function Access Control Metrics (MFACM), Data Member Access Control Metrics (DMACM), and Member Access Control Factor Metrics (MACF). Among these attribute related metrics, the AHF complexity metric of Abreu et al. is taken for formulating the Cognitive Weighted Attribute Hiding Factor (CWAHF) [12]. This article validates this newly proposed CWAHF complexity metric.

The following section 2 explains the mathematical formula for AHF and CWAHF. In section 3 three case studies are done to ensure the applicability of the metric. Section 4 theoretically validates the metric against Weyuker's nine properties and section 5 checks the validity with Abreu's seven criteria. Section 6 inspects the Kitchenham's conditions for validity. Section 7 deals with empirical and comparative study to check the repeatability and accuracy, along with the statistical validation of the complexity metric. Section 8 draws the conclusion and possible future works.

2. THE COMPLEXITY METRIC FOR VALIDATION

2.1. The Validation of Software Metrics

Software metric validation is a gruelling task due to lack of theoretical, empirical methodologies, and no agreed upon validation procedures to analyse the metric quality [13]. But the success of a metric depends critically on empirical, theoretical and statistical validations of software metrics[14]. Main aim of the software metric validation is to assess whether the metric is possessing the basic necessary properties of a valid measure and satisfies the sufficient conditions for valid metric.

There are many ways to do this. Fenton suggests that a valid metric must obey the representation condition of measurement theory, so that *intuitive* understanding of the attribute is preserved in mapping to a numerical relation system [15]. One way of validating is to experiment the complexity metric with different case studies to prove the applicability of the metric [16]. Another way of validating is to check whether, the metric satisfies various theoretical conditions posed by different validation frameworks like Weyuker's nine properties, or Abreu's seven criteria, or Kitchenham's four direct and four indirect properties for sound metric[3][17-18]. Empirical validation is yet another way. It involves empirical data collection and analysis to find out the veracity of the metric [19]. While theoretical methods of validation prove the validity of the measure with respect to some defined axioms, the empirical methods provide corroborating evidence of validity. One moreway is to check how the metric accurately represent those attributes they purport to quantify, [14] and the statistical correlation coefficient analysis can reveal the accuracy. Another common approach taken to validate new metrics is to show that they correlate with some well-known existing metrics [16-18]. Also, the evaluation of effective prediction of particular software quality aspect like maintainability, comprehensibility etc., can be used for validating the software complexity metric [20]. Some of these validation methodologies will be applied in the forthcoming sections of this article.

2.2. The CWAHF Complexity Metric

The complexity metric CWAHF, which is going to be validated, is derived from AHF complexity metric, which considers only the structural mechanism of the OO paradigm and does not include the cognitive aspect of the complexity. But Wang et al, observes that such traditional measurements cannot actually reflect the real complexity of software systems, and the ideal measure should include the cognitive complexity also, as it represents the real

semantic complexity by integrating both the operational and architectural complexities [21]. So, the CWAHF adds the cognitive complexity to the AHF metric. Francis Thamburaj and Aloysius have already derived the equation and calibrated the cognitive weights(CW) [12]. According to that the cognitive weight for private attribute is 2; for the default attribute is 3; for the protected attribute is 4 and for the public attribute is 1. For convenience, the formula for CWAHF is reproduced here.

$$AHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_h(C_i) + A_v(C_i)} \tag{1}$$

where $A_h(C_i)$ = Number of hidden attributes in class C_i; $A_v(C_i)$ = Number of visible attributes in class C_i; TC = Total number of Classes in the whole system. The formula for CWAHF metric is

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

where,

$$\sum_{i=1}^{TC} A_h(C_i) = \sum_{i=1}^{TC} A_p(C_i) * CW_{pa} + A_d(C_i) * CW_{da} + A_t(C_i) * CW_{ta}$$

$$\sum_{i=1}^{TC} A_v(C_i) = \sum_{i=1}^{TC} A_u(C_i) * CW_{ua}$$

$A_p(C_i)$ = Number of private attributes in class C_i; $A_d(C_i)$ = Number of default attributes in class C_i; $A_t(C_i)$ = Number of protected attributes in class C_i; $A_u(C_i)$ = Number of public attributes in class C_i; CW_{pa} = CW of private attribute; CW_{da} = CW of default attribute; CW_{ta} = CW of protected attribute; CW_{ua} = CW of public attribute; TC = Total number of Classes in the whole system.

3. EXPERIMENTATION AND CASE STUDIES

In this section, the applicability of the cognitive weighted complexity metric is experimented with three different case studies. The first case study checks the lower boundary and the second case study checks for the upper boundary of the cognitive complexity metric value. The third case study checks the middle level cognitive complexity metric value.

```
/******* Case Study 1 *******/
class C1{
    public int i1 = 10;
    public double d1 = 44;
    public float f1 = 3.3f;
    ….
}
class C2 {
    public int i2 = 20;
    public float f2 = 7.7f;
    public String str1 = "Francis";
    public double d2 = 3.0, d3 = 2.0;
    ….
}
```

```
/******* Case Study 2 *******/
class C3{
    private int i1 = 10;
    private double d1 = 44;
    private float f1 = 3.3f;
    ….
}
class C4 {
    private int i2 = 20;
    private float f2 = 7.7f;
    private String str1 = "Francis";
    private double d2 = 3.0, d3 = 2.0;
    ….
}
```

```
/******* Case Study 3 *******/
class C5{
    private int i1 = 10;
    public double d1 = 44;
    protected float f1 = 3.3f;
    ….
}
class C6 extends C5 {
    private int i2 = 20;
    protected float f2 = 7.7f;
    private String str1 = "Francis";
    double d2=3.0, d3 = 2.0;
    ….
}
```

In case study 1, all the attributes are public and class C1 has 3 attributes and class C2 has 5 attributes. The complexity metric value as per Eq. (1), AHF = (0+0) / (3+5) = 0. The cognitive weighted complexity metric value as per Eq. (2), CWAHF = (0+0) / (3*1 + 5*1) = 0. Hence, this case study proves that both the AHF and

CWAHF complexity metric values are 0 since all the attributes are public in scope. In other words, there is no hiding of any attribute, making the attribute hiding factor as 0. Thus the experiment with the lower boundary case is validated.

In case study 2, all the attributes are private and class C1 has 3 attributes and class C2 has 5 attributes. The complexity metric value as per Eq. (1), AHF = (3+5) / (3+5) = 1. The cognitive weighted complexity metric value as per Eq. (2), CWAHF = (3*2 + 5*2) / ((0*1+0*1) + (3*2 + 5*2)) = 1. Hence, the second case study proves that both the AHF and CWAHF complexity metric values are 1 since all the attributes are private in scope. In other words, all the attribute are hidden in each class, making the attribute hiding factor as 1. Thus the experiment with the upper boundary case is validated. Further, CWAHF complexity metric values remains within the range of [0, 1].

In case study 3, the attributes are mixed type. The class C1 has 1 private attribute, 1 public attribute, 1 protected attribute. The class C2 has 2 private attributes, 2 default attributes, 1 protected attribute. The complexity metric value as per Eq. (1), AHF = (2+5) / (1) + (2+5) = 7/8 = 0.875. The cognitive weighted complexity metric value as per Eq. (2), CWAHF = ((1*2 + 1*4) + (2*2 + 2*4 + 1*3)) / (1*1) + ((1*2 + 1*4) + (2*2 + 2*4 + 1*3)) = 21/22 = 0.955. The complexity metric value of CWAHF is greater than AHF, because of the cognitive load added to the AHF. Thus, case study 3 validates the complexity metric between the boundary values.

So, the case studies prove the practical applicability and dimensionless of the CWAHF metric.

4. VALIDATION BY WEYUKER'S PROPERTIES

Weyuker has proposed nine properties to evaluate any software complexity measure [22]. A good software complexity metric should follow all or at least the majority of these properties in order to be a good complexity metric, since they evaluate the weaknesses of a measure in a concrete way. Chidamber and Kemerer refer Weyuker's properties to validate their metric suite [23]. This section validates CWAHF against these criteria to establish the robustness of the complexity metrics.

Property 1: $(\exists P)\ (\exists Q)\ |\ (|\ P\ | \neq |\ Q\ |)$ *where P and Q are program bodies.*It states that complexity metric should not rank all program bodies as equally complex. The CWAHF has different complexity values in the case studies conducted in section 3. Hence it satisfies the first property.

Property 2: Let c be a non-negative number. Then there are only finitely many programs of complexity c. It says that there should be only a finite number of classes with same complexity metric value. Since, any software system will have only finite number of classes with same number and type of attributes, CWAHF satisfies this property.

Property 3: There are distinct program P and Q such that $|\ P\ | = |\ Q\ |$. It states that a metric should not be too "fine" to assign to every program a unique complexity. When different programs with same number and type of attributes will have same CWAHF value. So, this property is satisfied.

Property 4: $(\exists P)\ (\exists Q)\ |\ (P \equiv Q\ \&\ |\ P\ | \neq |\ Q\ |)$ *where P and Q are program bodies.*When the implementation style differs for functionally similar classes, the number and type of attributes may change resulting in different complexity values for CWAHF. Hence this property is satisfied.

Property 5: ($\forall\ P$) (Q) ($|\ P\ | \leq |\ P; Q\ |$ and $|\ Q\ | \leq |\ P; Q\ |$). This property is satisfied as the number of attributes for each class is less than the total number of attributes for all classes. Thus, the CWAHF follows the rule of monotonicity.

Property 6: $(\exists P)\ (\exists Q)\ (\exists R)\ |\ (|\ P\ | = |\ Q\ |)\ \&\ (|\ P; R\ | \neq |\ Q; R\ |)$; $(\exists P)\ (\exists Q)\ (\exists R)\ |(|\ P\ | = |\ Q\ |)\ \&\ (|R; P\ | \neq |\ R; Q\ |)$ As the CWs of CWAHF complexity metric are fixed, joining any new program to the two different classes adds the same amount of complexity to each of the class. Therefore, the property six is NOT satisfied.

Property 7: ($\exists P$) ($\exists Q$) *such that Q is formed by permuting the order of the statements of P and* ($| P | \neq | Q |$). Changing the order of attribute declaration and initialization in a class will not change the complexity metric values of CWAHF. Therefore, this property is NOT satisfied.

Property 8: If P is renaming of Q, then ($| P | = | Q |$). The complexity metric values of CWAHF are not affected by program name change obviously. So, this property is satisfied.

Property 9: ($\exists P$) ($\exists Q$) | ($| P | + | Q |$) < (P; Q). The last property states that the complexity of a program formed by concatenating two programs may be greater than the sum of their complexities. The complexity metric CWAHF depends on the hidden and total number of attributes. When two classes with identical types and number of attributes, their complexity values will be the same. When they are combined, some attributes may become redundant and will be removed. Hence, the complexity value of the combined program will be reduced. Therefore, this property is satisfied.

Thus, all the properties of Weyuker, except 6 and 7 are satisfied by CWAHF. Property 6 is not satisfied due to the independence of the number and type of attributes from the interaction involved in the software system. Property 7 does not fit to any OO software system. The high number of satisfied properties proves that CWAHF is theoretically sound and valid complexity metric.

5. VALIDATION BY ABREU'S CRITERIA

A set of seven criteria for evaluation of an OO complexity metric is proposed by Abreu *et al* [4]. This section validates CWAHF against these criteria

*Criterion 1: Metrics determination should be formally defined.*As the CWAHF complexity metric is formally defined with an equation, this property is satisfied. Hence, the subjectivity in the complexity measurement is not possible.

Criterion 2: Non-size metrics should be system size independent. The case studies of CWAHF shows that it is applicable to both inheritance tree based project as well as other simple projects varying over types, sizes and structural complexities. So, this property is satisfied by CWAHF.

*Criterion 3: Metrics should be dimensionless or expressed in some consistent unit system.*The third criterion states that the subjective or artificial measurement units for the complexity metrics should be avoided, in order to escape from the possibility of misinterpretations. As the metric units of CWAHF is in ratio scale, it is dimensionless as given in this criterion.

*Criterion 4: Metrics should be obtainable early in the life-cycle.*Since the number and types of scope attributes can be collected in design phase itself, even though more accurate number can be got at the coding level.So, CWAHF satisfies this criterion and helps to reduce the cost and human effort in developing the software system.

*Criterion 5: Metrics should be down-scalable.*It talks about the applicability of complexity metric both in the system level as well as in the subsystem level. It is satisfied since the down-scalable criterion is applicable to CWAHF, as it is calculated in the class level and added for system level metric.

*Criterion 6: Metrics should be easily computable.*The sixth property deals with production cost reduction and automation of complexity metric collection. Because of finding and counting the different types of scope attributes can be done easily and CWAHF calculation involves only simple arithmetic, it can be easily computerized. Therefore, this criterion is satisfied.

*Criterion 7: Metrics should be language independent.*The independence of the language is satisfied since CWAHF can be applied to any OO languages with the specific bindings as Abreu et al. has done it for C++ [24] and Eiffel language [25].

Thus, all 7 criteria are satisfied by CWAHF. Hence, it is theoretically sound and valid metric.

6. VALIDATION BY KITCHENHAM

Kitchenham and Pfleeger proposed four criteria for direct metrics and in addition to that four more conditions for indirect metrics in 1995 [26]. The conditions for direct metrics are as follows:

1) *For an attribute to be measurable, it must allow different entities to be distinguished from one another*:It is related to Weyuker's first property, which states that the measure gives unique values for two different classes or entities. The CWAHF metric obviously gives different values for different classes, since each class may have different number of attributes and the scope may also vary. Hence this property is satisfied.

2) *A valid metric must obey the representation condition*:The metric must keep the intuitive notion about the attribute and how it differentiates different entities. This is connected with the eighth property of Weyuker which says that the renaming of the class or entity should not affect the value of the measure. This is very much true with respect to CWAHF, since it deals only with the number and scope type of variables in each class. Therefore, it is satisfied.

3) *Each unit of an attribute contributing to a valid metric is equivalent*:This property is not only specific to complexity measure, but applicable to all types of measures. It is related to third property of Weyuker that demands that metric should not be too fine-grained to yield unique value for all entities or classes. This is true in the case of CWAHF, since the complexity value will vary according the number and scope types of attributes in each class. So, it is satisfied.

4) *Different entities can have the same attribute value*: This property deals with the coarseness of the metric. The metric should allow to have same value for two different entities. In the case of CWAHF, two different classes with the same number of attributes and same types of scopes will have same complexity value. Hence, this condition is satisfied.

These four properties are necessary but not sufficient to cope with indirect measures. Therefore, Kitchenham puts forward four more additional conditions for indirect metrics.

5) *The metric should be based on an explicitly defined model of the relationship between certain attributes*:The relationship model is usually between the internal attributes like AHF, MHF and the external attributes like comprehensibility, maintenance, etc. Since AHF complexity metric with lower value promotes maintenance and modifiability, it satisfies this property. As the CWAHF is the extension of AHF, it also satisfies these external attribute relationship. Further CWAHF is proved to satisfy the external quality of comprehensibility with augmented cognitive weight factor.

6) *The model must be dimensionally consistent:*This is related to the third criteria, namely, 'metrics should be dimensionless' of Abreu et al. Dimensionless can be achieved by expressing the metric values in some consistent unit system. The measurement units of CWAHF is in the ratio scale and expressed in the range of zero to one. Hence it satisfies this criteria.

7) *The metric must not exhibit any unexpected discontinuities*:According to Harrison et al., in AHF metric, there is a discontinuity in the case of single class which will not be the usual case in systems and projects. Further, AHF deals only with the quantity and not the quality of data hiding [3]. So, AHF satisfies this condition. Consequently, CWAHF metric is also valid since it is nothing but AHF loaded with cognitive weights.

8) *The metric must use units and scale types correctly*: Al mathematical and statistical operations cannot be done on all scale types such as nominal, ordinal, and interval, ratio, and absolute. The CWAHF metric is in ratio scale. Therefore, all arithmetic operations can be meaningfully applied [27, p.58]. The calculation of CWAHF uses only addition, division, and multiplication. So, this criteria is satisfied.

Harrison et al. has already proved that AHF satisfies all the eight conditions taking into account of the facts that there is a discontinuity in the case of single class [3]. Now, CWAHF is also proved to satisfy all the eight conditions of Kitchenham et al.

7. COMPARATIVE STUDY & STATISTICAL ANALYSIS

In this section the comparative study of CWAHF is done against the AHF complexity metric in the widely accepted and empirically verified MOOD metric suite proposed by Abreu et al [4]. For this, online comprehension test was conducted to a group of forty students, with five programs, P1, P2, P3, P4, and P5, in Java. The completion time was captured and the average time taken was calculated and placed in Table. 1 under the column head Comprehension Mean Time (CMT). The complexity values of AHF and CWAHF are calculated manually for each of the five programs, using the Eq. (1) and Eq. (2), as shown in the case study and are placed in the column AHF and CWAHF. The pictorial representation of the Table. 1 is shown in the Fig. 1. The general trend of the comprehension mean time, in Fig. 1, is closer to the CWAHF values than the AHF values. This proves that CWAHF complexity metric is closer to reality and more accurate in nature. The higher metric values of CWAHF, in all cases, is due to the augmented cognitive weight factor.

Table 1
Attribute Complexity Values with CMT

Program	AHF	CWAHF	CMT
P1	0.923	0.973	345.47
P2	0.88	0.9565	325.68
P3	0.8	0.9167	357.92
P4	0.83	0.9333	295.91
P5	0.75	0.875	199.89

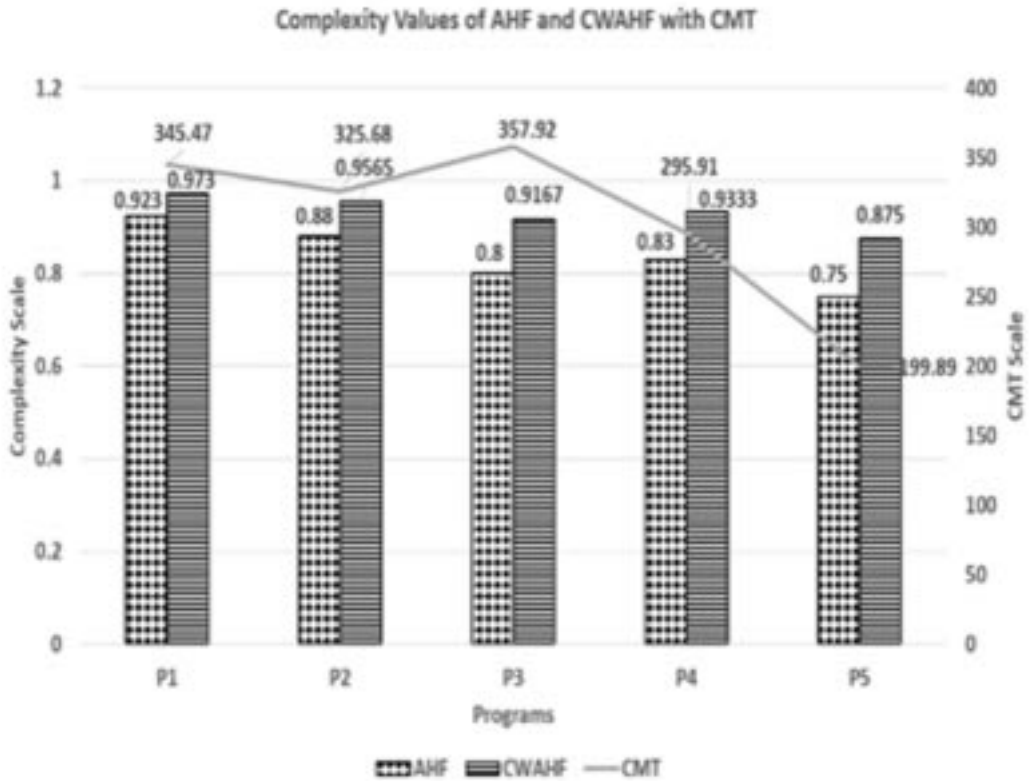

Figure 1: Comparative study of AHF and CWAHF with CMT

A correlation analysis was performed between AHF and CMT complexity values and also between CWAHF and CMT values. Both correlations were found to be positive, which means that both values of AHF and CWAHF correlates well with CMT values found in the empirical test conducted. The Pearson correlation r (AHF, CMT) is 0.6804 and r (CWAHF, CMT) is 0.7642. The bigger correlation value for CWAHF than the AHF concludes that CWAHF is a better indicator of complexity of the classes with various types of attribute hiding. Thus, CWAHF complexity metric is also statistically proved to be valid and more precise than AHF complexity metric.

8. CONCLUSION

The validation of CWAHF complexity metric is done using seven different methods. The applicability of the metric is proved with three different case studies. The theoretical evaluation against Weyuker's nine properties, Abreu's seven criteria,and Kitchenham's eight conditions, emphatically asserted that CWAHF complexity metric is a sound and robust metric. It is also empirically verified with a set of program comprehension tests to examine the repeatability. The comparative study done with AHF complexity metric has exhibited that CWAHF complexity metric is more accurate than AHF complexity metric.Finally, it is checked with the statistical analysis using the Pearson correlation which has proved CWAHF complexity metric is a better and more comprehensive indicator than the AHF complexity metric. All the seven validations proved the practical applicability, theoretical veracity, empirical repeatability and statistical accuracy of the complexity metric CWAHF beyond doubt.

As the future work, large scale empirical validation can be done over open source software systems. Also, validations can be done against specific aspect of the software quality like maintainability that plays a high priority role in software industry.

REFERENCES

[1] C. Zoller et al., "Measuring Inappropriate Generosity with Access Modifiers in Java Systems,"*22nd Int. Workshop on* **IWSM-MENSURA**, *IEEE*, 2012.

[2] Tahvildari et al., "Categorization of Object-Oriented Software Metrics,"*Proc. IEEE Canadian Conference on Electrical and Computer Eng*., 235–239, May 2000.

[3] Harrison, R., Counsel, S.J. and Nithi, R.V., "An Evaluation of the MOOD Set of Object-Oriented Software Metrics,"*IEEE Tr. on Software Eng.*, **6**, 491-496, 1998.

[4] F. B. Abreu et al., "Object-Oriented Software Engineering: Measuring and Controlling the Development Process,"*Proc. 4th Int.Conf. Software Quality*, **186**, 1-8, 1994.

[5] F. B. Abreu, "Using OCL to Formalize Object-Oriented Metrics Definitions,"*5th Int. ECOOP Workshop on Quantitative Approaches in OO Software Eng*., (**QAOOSE 2001**).

[6] Henderon-Sellers, B., "*Object-Oriented Metrics: Measures of Complexity*," Upper Saddle River, New Jersey, Prentice Hall, 1996.

[7] J. Bansiya and C.G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment," *IEEE Transactions on Software Engineering*, **1**, 4-17, 2002.

[8] S. Pasupathy and Bhavani R., "Object Oriented Metrics Evaluation,"*International Journal of Computer Applications*, **1**, 30-37, September 2013.

[9] V. Singh and Vandana Bhattacherjee, "A New Complete Class Complexity Metric,"*International Journal of Soft Computing and Software Engineering* (JSCSE), E-ISSN: 2251-7545, **9**, 1-9, September 2013.

[10] K.P. Srinivasan, T. Devi, "A Comprehensive Review and Analysis on Object-Oriented Software Metrics in Software Measurement,"*International Journal on Computer Science and Engineering* (IJCSE), ISSN: 0975-3397, **7**, 247-261, July 2014.

[11] R.S. Chhillar et al., "An Access Control Metric Suite for Class Hierarchy of Object-Oriented Software Systems,"*International Journal of Computer and Communication Engineering*, **1**, 61-65, January 2015.

[12] Francis Thamburaj and A. Aloysisus, "Cognitive Perspective of Attribute Hiding Factor Complexity Metric,"*International Journal of Engineering and Computer Science*, ISSN: 2319-7242, **11**, 14973-14979, November 2015.

[13] Jacquet, J-P., and Alain Abran, "From software metrics to software measurement methods: A process model,"*Software Engineering Standards Symposium and Forum, Emerging International Standards. **ISESS 97**., Third IEEE International*,128-135, 1997.

[14] K. P. Srinivasan, and T. Devi, "Software Metrics Validation Methodologies in Software Engineering,"*Int. Jour. of Software Eng. & Applications*,**6**, 87-102, 2014.

[15] N. Fenton, "Software Measurement: A Necessary Scientific Basis,"*IEEE Trans. Software Eng.*, **3**, 199-206, March 1994.

[16] Francis Thamburaj and Aloysius, A., "Cognitive Weighted Polymorphism Factor: A Comprehension Augmented Complexity Metric," WASET, *Int. Jr. of Computer, Electrical, Automation, Control and Information Engineering*, **11**, 1982–1987, 2015.

[17] Francis Thamburaj and A. Aloysius, "Validation of Cognitive Weighted Method Hiding Factor Complexity Metric,"*International Journal of Applied Engineering Research* (IJAER), Print ISSN: 0973, Online ISSN 1087-1090, **82**, 91-96, December, 2015.

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

[19] S. Misra, "An Approach for the Empirical Validation of Software Complexity Measures,"*Acta Polytechnica Hungarica*, **2**, 141-160, 2011.

[20] L. Kumar, D. K. Naik, and S. K. Rath, "Validating the Effectiveness of Object-Oriented Metrics for Predicting Maintainability,"*Scinece Direct Procedia Computer Science*, **57**, 798-806, 2015.

[21] Y. Wang, and J. Shao, "Measurement of the Cognitive Functional Complexity of Software,"*Proc. Second IEEE Int. Conf. Cognitive Informatics* (**ICCI'03**), 2003.

[22] Elaine J. Weyuker, "Evaluating Software Complexity Measures", *IEEE Transactions on Software Engineering*, **9**, 1357-1365, Sep. 1988.

[23] Chidamber S.R., Kemerer C.F., "Towards a Metrics Suite for Object-Oriented Design,"*Object-Oriented Programming Systems, Languages and Applications* (OOPSLA), **26**, 197–211, 1991.

[24] F. B. Abreu et al., M. Goulao, and R. Estevers, "Toward the Design Quality Evaluation of Object-Oriented Software Systems,"*Proc. Fifth Int. Conf. Software Quality*, 1995.

[25] F. B. Abreu et al., "The Design of Eiffel Programs: Quantitative Evaluation Using the Mood Metrics,"*In Proceedings of **TOOLS'96***, California, July, 1996.

[26] Kitchenham, B., Pfleeger, S.L., and Fenton, N., "Towards a Framework for Software Measurement Validation,"*IEEE Tr. on Software Eng.*, **12**, 929-943, 1995.

[27] Norman Fenton, and James Bieman, "*Software Metrics: A Rigorous and Practical Approach*," 3rd Ed., CRC Press, 2014.